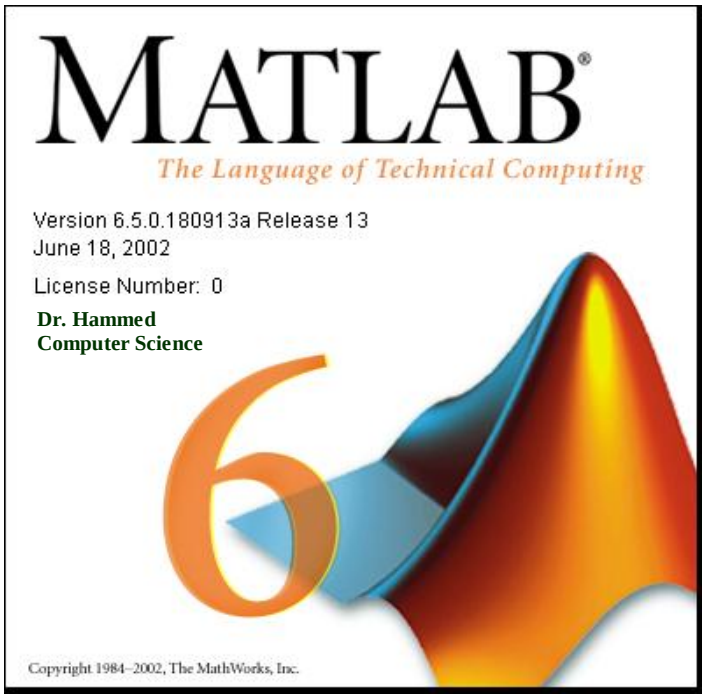
وزارة التعليم العالي والبحث العلمي
قسم علوم الحاسبات

البرمجة بلغة



إعـداد

مجموعة من التدريسيين لمادة علوم الحاسبات

نيسان ٢٠٠٨

لغة البرمجـــــــــة MATLAB:

(The MATLAB programming language)

~~~~~~

١- مقدمة عن لغة الاحتساب التقني MATLAB.

٢- الثوابت والمتغيرات.

٣- المصفوفات والعمليات على المصفوفات.

٤- المصفوفات متعددة الأبعاد.

٥- مصفوفات الخلايا.

٦- السلاسل الرمزية.

٧- جمل الإدخال والإخراج.

٨- الجمل الشرطيـــــة.

٩- جمل الدوران والتكرار.

١٠- ملفات البيانات الخاصة ببرنامج MATLAB.

١١- ايعازات المجموعات والبتات والايعازات القاعدية.

١٢- الدوال والبرامج الفرعية.

١٣- الرسوم البيانية.

المصـــــــــادر:

١- MATLAB 6.5 الدليل المرجعي والتعليمي، المهندس عبد الكريم البيكو، (دار شعاع للنشر).

٢- MATLAB Help Version 6.5
~~~~~~

٣- www.mathworks.com

لغة البرمجة MATLAB : The MATLAB Programming Language

مقدمة

يعتبر برنامج MATLAB البرنامج الأشهر في الأوساط العلمية، إذ يستخدم هذا البرنامج في معظم المسائل العلمية والهندسية، وبعد نمذجة أي مسألة أو ظاهرة يأتي بعدها دور هذا البرنامج ليتعامل مع تلك البرامج ويحللها بأبسط الطرق وأحدثها وأيسرها برمجة، ومن الجدير ذكره بان هذا البرنامج يعلّم أكثر من ٢٠٠ معهد وكلية في الولايات المتحدة الأمريكية فقط، عدا تلك المعاهد في أوربا وبقية العالم، ويكفي أن تدخل إلى أحد محركات البحث على شبكة الانترنت وتكتب فقط MATLAB، فستُذهل من عدد المواقع التي تتحدث عن هذا البرنامج.

وتعتبر لغة MATLAB لغة برمجية عالية الأداء تستخدم لإجراء الحسابات التقنية، وتقوم بعمليات الحساب والإظهار ضمن بيئة سهلة البرمجة كما أنها لا تحتاج إلى احتراف كبير. تمكنك هذه اللغة من حل العديد من المسائل التقنية حسابيا، خاصة التي يعبر عنها بمصفوفات والتي تحتاج إلى جهد كبير لبرمجتها بلغات البرمجة الأخرى مثل لغة C و FORTAN.

أتت تسمية هذه اللغة من اختصار التعبير **MATrix LABoratory** (مختبر المصفوفة)، حيث إن البرنامج مصمم أساسا للتعامل مع العمليات على المصفوفات بشكل بسيط. كما أرفقت بهذه اللغة أدوات لمعالجة وحل تطبيقات علمية خاصة سميت toolboxes (وهي أكثر من عشرين أداة)، وتعتبر هذه الأدوات هامة جداً لمستخدمي هذه اللغة، حيث تسمح لهم بتعلم وتطبيق تقنيات حل متخصصة لمعالجة مشكلات ومسائل خاصة، مثل معالجة الإشارة، ونظم التحكم والمحاكاة والشبكات العصبية والتحليل الكمي والمالي والإحصاء ومسائل الجبر الخطي والامثلية ... الخ.

يؤمن برنامج MATLAB أدوات واجهة التخاطب الرسومية Graphical User Interface (GUI) التي تجعلك تتعامل مع البرنامج على انه أداة تطبيقية متطورة.

تشغيل برنامج MATLAB

يتم تشغيل البرنامج بأحد الطرق التالية:

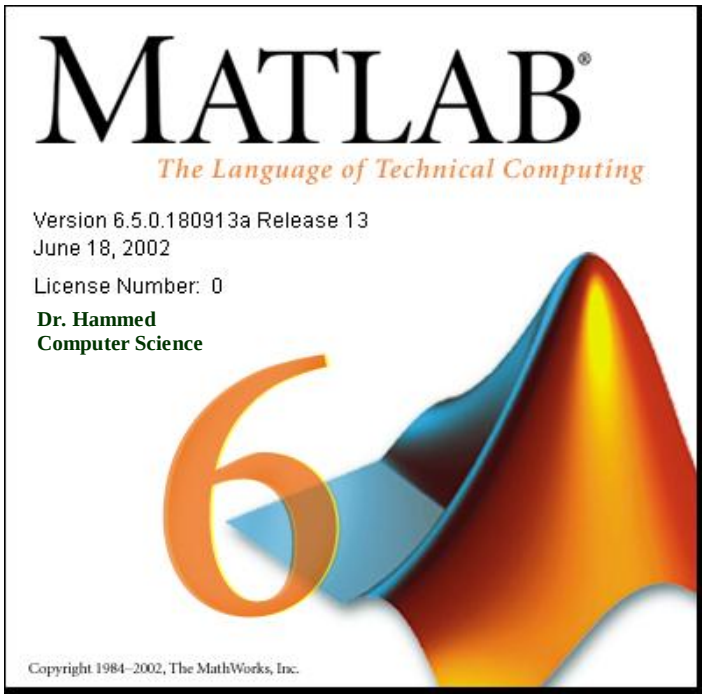
١- بعد تنصيب برنامج MATLAB على الحاسبة التى تعمل عليها. يتم إضافة رمز أيقونة البرنامج على سطح مكتب الحاسبة ويحمل الرمز 

ويتم فتحة عند النقر على الأيقونة بنقرتين مزدوجتين double click.

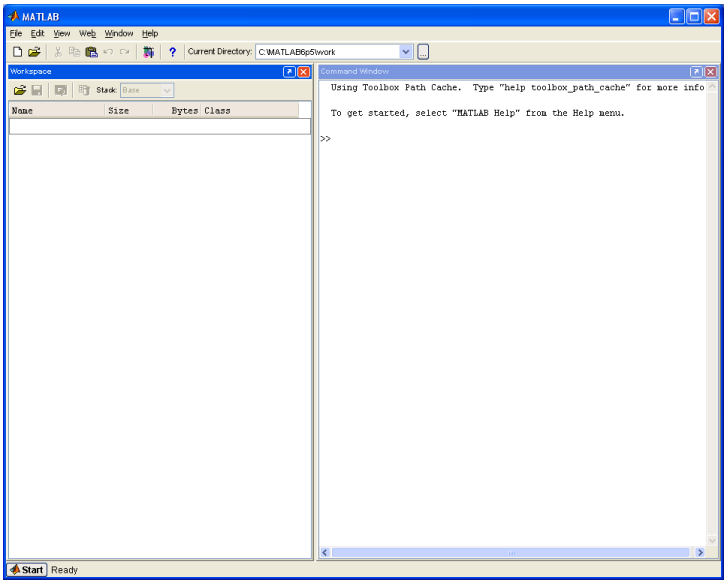
٢- أو عن طريق الذهاب إلى قائمة start ومنها إلى برامج Programs ثم أسم البرنامج MATLAB 6.5.

start ⟵ Programs ⟵ MATLAB 6.5

عندها سوف تظهر لنا شاشة تحمل أسم البرنامج MATLAB ونسخة الإصدار وسنة النشر كما في الشكل رقم (١). ثم بعد ثواني قليلة تظهر نافذة البرنامج الرئيسية والتي تكون في بداية التشغيل كما في الشكل رقم (٢) حيث تحتوي هذه النافذة كسائر البرمجيات التي تعمل تحت بيئة نظام Windows على نوافذ فرعية.



شكل (١): شاشة اسم البرنامج MATLAB



شكل (٢): شاشة نافذة البرنامج الرئيسية (سطح مكتب MATLAB)

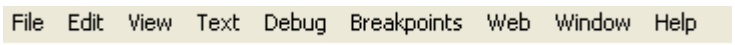
سطح مكتب برنامج MATLAB

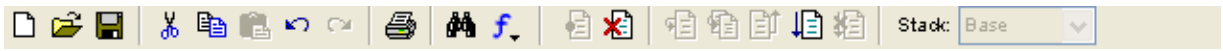
عند تشغيل برنامج MATLAB ستظهر على شاشتك عدة نوافذ عنوان احدها MATLAB وتسمى سطح مكتب برنامج MATLAB، تحوي هذه النافذة وتتحكم بجميع النوافذ الأخرى المكونة لبرنامج MATLAB. وحسب خيارات تنصيب البرنامج، فقد تكون بعض هذه النوافذ مرئية أو مخفية ضمن نافذة MATLAB.

مكونات نافذة MATLAB

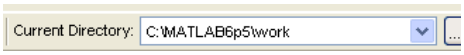
تتكون نافذة MATLAB من الأجزاء التالية:-

١- شريط العنوان ويكون ذات لون مميز عن باقي الأشرطة يوجد على يساره الرمز الصوري للبرنامج وأسم البرنامج 

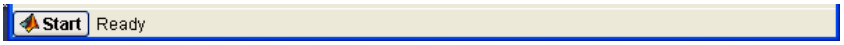
٢- شريط قوائم (Menu Bar) أو (Lists Bar) يبدأ بقائمة ملف File، قائمة تحرير Edit، قائمة عرض View، ... وحتى قائمة المساعدة Help. 

٣- شريط الأدوات (Tools Bar) ويضم رموز صورية لبعض الايعازات الموجودة في قوائم الشريط السابق. 

هناك في الجزء الأخير من شريط الأدوات جزء مهم يدعى الدليل الحالي (Current Directory) والذي يخبر المستخدم في أي جزء من الحاسب هو موجود حالياً وكما في الشكل (٢) يعلمنا بأننا على الدليل (المجلد) MATLAB6P5\work وعلى القرص C:



٤- هنالك شريط مهام خاص بنافذة برنامج MATLAB وفيه كلمتان الأولى Start وعملها كطريق مختصر لتنفيذ بعض الايعازات. بينما Ready تعلمك بأن البرنامج جاهز للعمل حسب التوجيه المعطى له.

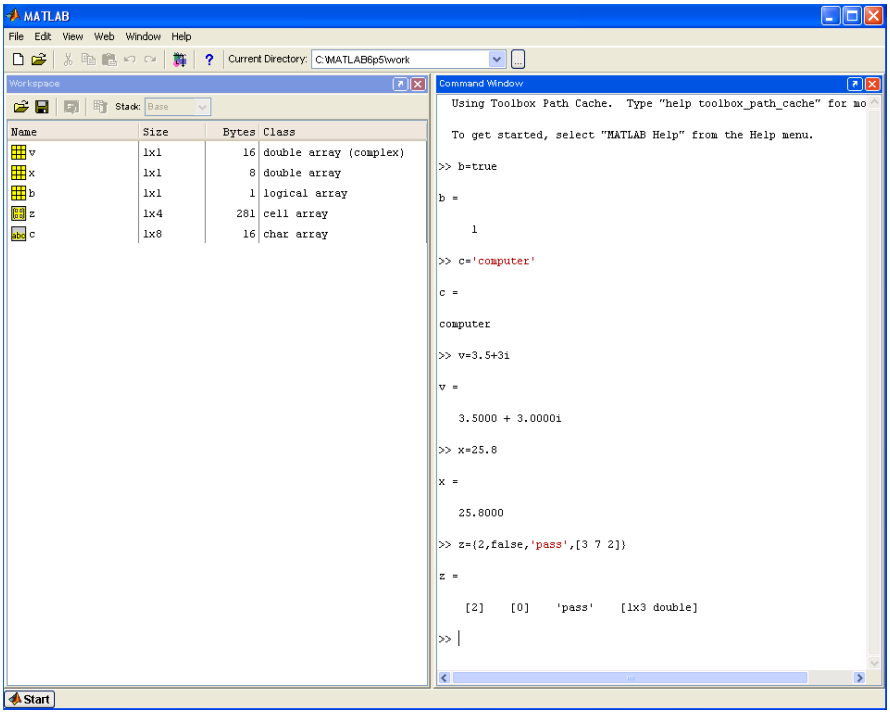


بالإضافة إلى الأشرطة أعلاه هناك مجموعة من النوافذ الفرعية التي يمكن تفعيلها أو إخفائها حسب الحاجة وذلك كما في الشكل (٣) حيث يتم تأشير أسم النافذة المرغوب بعرضها بإشارة (√)، لكن هناك نافذة أساسية للعمل هي نافذة الأمر Command Window، والتي من خلالها يتم التعامل بكتابة وتنفيذ الأوامر بصورة مباشرة أو غير مباشرة.

٥- تعتبر النوافذ الداخلية الظاهرة أسمائها في قائمة View كما في الشكل رقم (٣) هي من مكونات نافذة برنامج MATLAB ولكل نافذة منها عملها الخاص وكما يلي:-

أ- نافذة الأمر Command Window: وهي نافذة لا يمكن الاستغناء عنها لأن بواسطتها يتم تنفيذ الأوامر وعرض النتائج التي نحصل عليها من تنفيذ تلك الأوامر وتكتب بعد علامة الحث (<<).

ب- نافذة ساحة العمل Workspace: وهي عن واجهة تخاطبية تسمح لك باستعراض وتحميل وحفظ متغيرات لغة MATLAB حيث تظهر قائمة تضم أسم المتغير وحجمه وعدد بياناته وصنفه (جميع متغيرات لغة MATLAB هي من صنف مصفوفة)، كما في الشكل (٤).



شكل (٤): نافذة الأمر Command Window ونافذة ساحة العمل Workspace.

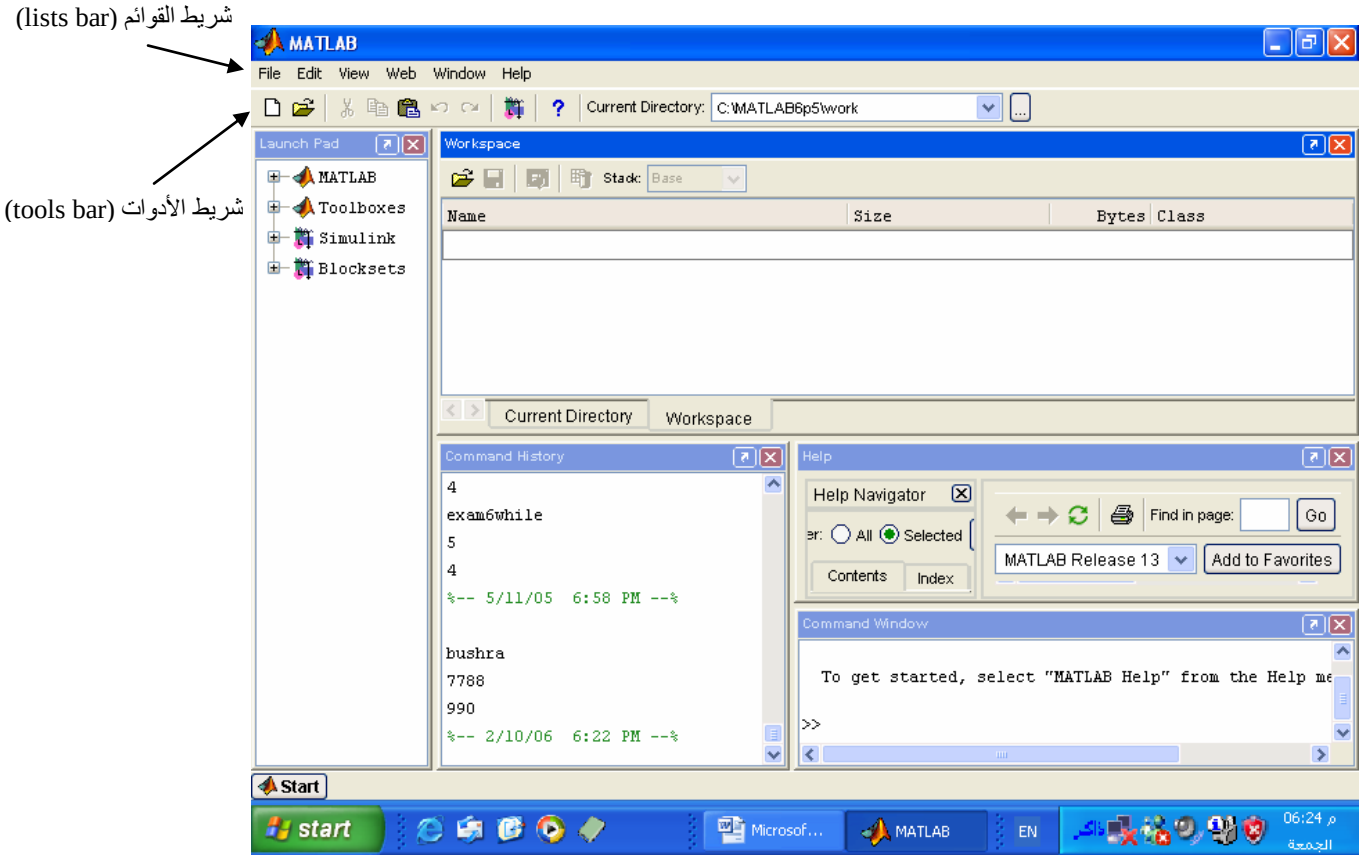
ج- نافذة الدليل الحالي Current Directory: وهي أيضا واجهة رسومية تحدد الدليل الحاوي للملف الذي يتعامل معه برنامج MATLAB.

د- نافذة المساعدة Help: وهي نافذة تخاطبية (رسومية) تسمح لك بالبحث واستعراض الوثائق بشكل مباشر.

و- لوحة البرامج التنفيذية Launch Pad: وهي عبارة عن نافذة تستعرض بنية شجرية للأدوات والبرامج التنفيذية.

هـ- نافذة الأوامر السابقة Command History: تمكنك هذه النافذة من إعادة تنفيذ الأوامر السابقة المنفذة في نافذة الأمر بدلاً من كتابتها مرة أخرى.

والشكل (٥) يبين النوافذ الداخلية لنافذة البرنامج MATLAB بعد تفعيلها ....



شكل (٥): النوافذ الداخلية لنافذة البرنامج MATLAB بعد تفعيلها

ملاحظات:

١- كل متغير في MATLAB عبارة عن مصفوفة، لغة MATLAB موجهة بالمصفوفات حتى وان كانت المتغيرات أعداداً مفردة (scalar).

٢- الأمر clear ضمن Workspace يستخدم لحذف المتغيرات والدوال من الذاكرة .

٣- الأمر clc ضمن Workspace يستخدم لمسح نافذة الأمر Command Window.

٤- يمكن إظهار النتائج العددية في لغة MATLAB بتنسيقات أظهار مختلفة:

format short, long, short e, long e, hex, blank, +, rat,…

٥- يمكن تنفيذ demo (demonstration) كايعازات جاهزة في MATLAB من خلال:

Help ⟵ Demos ⟵ أختر الموضوع المحدد (مثلا Toolboxes)

يمكن الاستفادة منها في:-

أ- تنفيذ Demos. ب- تعليم أكثر حول الموضوع (Help). ج- عرض شفرة البرنامج . د- استنساخ شفرة البرنامج من Demo إلى M–file.

ملاحظـــة : يمكن الدخول إلى Demos من Start

٦- ثلاث نقاط متتالية (...) في نهاية السطر مسبوقة بفراغ للدلالة على استمرار الايعاز في السطر التالي.

٧- فارزة منقوطة بعد الإيعاز تمنع طباعة المتغير أو الناتج في نافذة Command وكذلك في نافذة Editor.

٨- إشارة النسبة المئوية (%) تستخدم للتعليق فكل نص يأتي بعدها يعتبر نص تعليق، مثل:

% This Program Compute Area

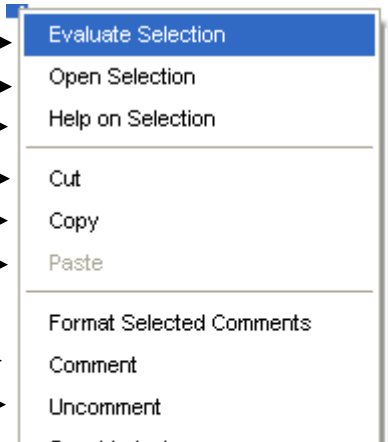
٩- ملفات MATLAB تسمى M–files وتكون توسعها (.m)، مثلاً (examplel.m).

١٠- الاحتفاظ بكتابة الايعازات السابقة واللاحقة في نافذة Command بحركة السهم للأعلى والأسفل.

١١- نتيجة تنفيذ برنامج MATLAB (النتائج والاخراجات) تظهر في شاشة Command Window لذلك يجب الانتقال إليها بعد التنفيذ.

١٢- لغة MATLAB لا تحتاج إلى الإعلان عن المتغيرات والثوابت والأنواع البيانية الأخرى المستخدمة بالبرنامج.

١٣- لتنفيذ مقطع من البرنامج (تعليمة أو مقطع من البرنامج) يتم تأشيرها ثم النقر أيمن فتظهر القائمة المنسدلة:



لحساب الجزء المؤشر وإظهار النتيجة في نافذة Command
الذهاب إلى الدالة المؤشرة
الذهاب للـ Help للجزء المؤشر
قص
نسخ
لصق
تعليق
رفع التعليق
هيكلة المقطع

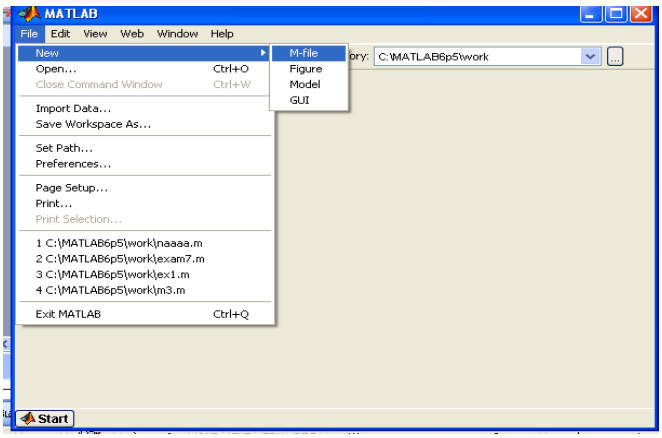
وبعد ذلك نختار Copy وننتقل إلى Command Window ونختار Paste وننفذها.

١٤- لإنشاء ملف نصي M-file، أنقر على أيكونة الصفحة الفارغة (البيضاء) الموجودة ضمن شريط أدوات سطح مكتب MATLAB، أو اختر New من القائمة File ومن ثم اختر M-file أو اختيار Open لفتح ملف موجود مسبقا. يستدعي هذه الأوامر نافذة محرر النصوص التي يمكنك في كتابة أوامر MATLAB (نافذة كتابة البرامج). كما في الأشكال (٦)، (٧)، (٨).

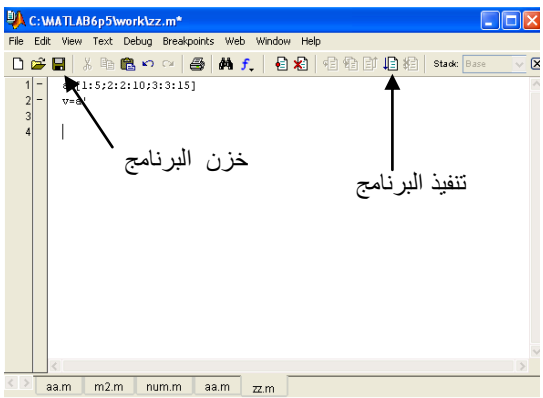
١٥- يمكن تنفيذ الملف المخزون باختيار أيكونة Run الموجودة في شريط أدوات نافذة Editor أو عبر ضغط المفتاح F5 أو الاختيار Run من القائمة Debug، أو كتابة اسم الملف المخزون أمام علامة الحث >> في نافذة Command. بعد انتهاء كتابة البرنامج (الملف) يخزن هذا الملف كملف M-file باسم معين (مثلاً example1.m) على قرصك الصلب عبر اختيار الاختيار Save من القائمة File أو الخزن ضمن شريط أدوات سطح مكتب MATLAB). كما في الشكل (٩).

ملاحظة:

عند الخزن يحمل البرنامج اسم افتراضي (Untitle1) فبالإمكان إعطاء الاسم الذي يرغب فيه المبرمج أو البقاء عليه.



شكل (٦): إنشاء ملف جديد.



شكل (٧): نافذة كتابة البرامج (محرر الملفات النصية).

١٦- عندما نريد إغلاق برنامج MATLAB عبر الاختيار Exit MATLAB من القائمة File الموجودة في نافذة سطح مكتب MATLAB أو عبر كتابة الأمر Exit في نافذة Command، أو علامة (×) في زاوية سطح مكتب MATLAB العليا اليمنى.

١٧- الإيعاز global للإعلان عن متغير عالمي بين الدوال والبرنامج الرئيسي.

١٨- لحساب زمن تنفيذ البرنامج نضع تعليمتي (tic) و (toc) بين الايعازات.

مثال:

```
clc;
clear;
tic;
(commands)
t = toc;
```

يفضل كتابتها في بداية أي برنامج رئيسي

١٩- للبحث عن إيعاز في الـ Help من خلال:

Help ⟶ MATLAB Help

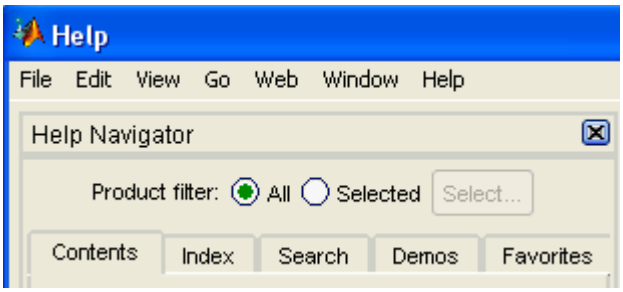
هناك عدة طرق للبحث عن الإيعاز، منها:-

1- Contents.

2- Index.

3- Search.

4- Demos.



٢٠- الإيعاز break يقوم بإيقاف تنفيذ البرنامج أو جزء من البرنامج أو الدالة (التعليمات التي بعد break لا تُنفذ).

أمثلة:

```
.....                  if .....               .....
.....                    .....                .....
.....                  else                   .....
.....                    break;               break;
  break;               {else توقف في حالة}    .....
{توقف تنفيذ الدالة فقط}                       .....
```

لا تنفذ

٢١- الإيعاز return للخروج من الدالة.

مثال:

```
function .....
```

……..

…….

……

return; خروج من الدالة

……..

……..

لا تنفذ

٢٢- رسالة الخطأ تحتوي على رقم السطر ونوع الخطأ.

٢٣- أي عملية حسابية غير منسبة إلى متغير تنسب تلقائيا إلى المتغير ans.

٢٤- أي متغير غير مستخدم ويدخل في العمليات فان البرنامج سوف يعطي خطأ.

رموز لغة MATLAB : MATLAB Symbols

تتكون لغة MATLAB من العناصر الأساسية التالية:

أ- حروف أبجدية إنكليزية: وهي: A, B, … , Z, a, b, … , z

ب- أرقام حسابية: 0, 1, 2, …, 9

ج- رموز خاصة مثل:) , (, * , ; , < , > , =, - , + ,{ , ... الخ.

الثوابت Constants:

يوجد في لغة MATLAB أنواع متعددة من الثوابت أهمها:-

(أ) الثوابت العددية Numerical Constants:

وتتكون من عدد من الأرقام ولها عدة أشكال هي:

(١) الثوابت الصحيحة: مثل: 0, +23, 472, -18

ملاحظة: أكبر عدد صحيح مستخدم.

```
>> bitmax
ans =
  9.007199254740991e+015
```

والتي تقابل $2^{53}-1$

(٢) الثوابت الحقيقية: مثل: 0.0, 51.8, 472.5, -18.0

ملاحظة:

```
>> realmin
ans =
  2.225073858507201e-308
>> realmax
ans =
  1.797693134862316e+308
>> pi
ans =
  3.146
```

(٣) الثوابت الحقيقية المدونة تدويناً يائياً: حيث تحول الصيغة الجبرية 10^N إلى صيغة MATLAB يائية EN فمثلا تصبح 2.0×10^3 في الجبر: 2.0E3 أو 2.0E+3 بالتدوين اليائي في MATLAB وكذلك تصبح 1.7×10^{-2} في الجبر: 1.7E2- في التدوين اليائي وكذلك تصبح 0.0032 :

3.2×10^{-3} : 3.2E-3

(٤) الثوابت العقدية: مثل: 1 - 2i ، 6 - 9i ، 6 + sin (0.5) * j ، sqrt (-2)

حيث: $i = j = \sqrt{-1}$

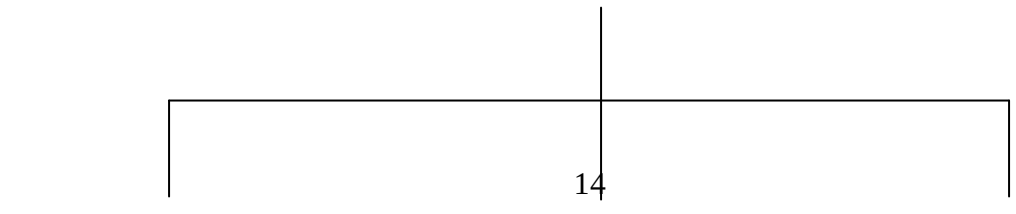
مثال ١: إذا كان: c = -7.7782 – 4.9497i

فلاستخراج الجزء الحقيقي cr = real (c) ⟸ cr = -7.7782

ولإستخراج الجزء التخيلي ci = imag (c) ⟸ ci = -4.9497

مثال ٢: c2 = 3 * (2 – sqrt (-1) * 3) ⟹ 6.000 – 9.000i

Numeric الرقمية (العددية)



مثال ١:

```
>> x = 100;
>> x = uint8 (x);
>> y = x + 1;
Error
```

مثال ٢:

```
>> x = 100;
>> x = double (x);
>> y = x + 1;
   y  =    101
```

(ب) الثوابت الرمزية String Constants:

يسمى هذا النوع من "ثوابت" مجازاً لأن الثابت هذا يتكون من حروف وأرقام ورموز توضع بين علامتي اقتباس quotations مفردة أي ' ' ويستخدم عادة كعناوين توضح القيم الناتجة من الحسابات ووحداتها، تسمى العبارات التالية والموجودة بين الحاصرات العليا ثوابت رمزية.

'The speed of wind ='

'I love Basrah'

'My birthday = 1970'

كل الثوابت الرمزية أعلاه، وان استخدمت أرقاما حسابية داخلها، فهي لا تحمل معنى حسابي، ومن الجدير بالذكر أثناء استعمال الثوابت الرمزية انه لا يجوز استخدام حاصرات علوية داخل حاصراتها، كما ينبغي التنبيه أي أن هناك قيماً رمزية للحروف يعتبر الحرف A اقل من الحرف B ويمكن كتابة ذلك بالصورة:

'A' < 'B'

(جـ) الثوابت المنطقية Boolean Constants:

وهي الثوابت التي قيمتها العددية (1) في حالة true و (0) في حالة false.

مثال:

$3 > 2 \quad \Longrightarrow \quad 1$

$0 > 5 \quad \Longrightarrow \quad 0$

المتغيرات Variables:

هناك بعض القواعد الواجب مراعاتها عند كتابة اسم المتغير وهي:

١. لا يمكن استخدام الكلمات المفتاحية (الكلمات المحجوزة) أو الدوال التي توفرها اللغة كأسماء متغيرات، مثال:

if, end, for, break, else, global, return, function, sin, log, …

٢. أسماء المتغيرات حساسة لحالة الحرف (COST, CoST, cost, Cost متغيرات مختلفة، وكذلك A و a).

٣. حرف l (small letter) في لغة MATLAB يشبه رقم 1.

٤. يمكن لأسماء المتغيرات أن تحوي 63 رمزا وسيهمل أي رمز زائد عن 63.

٥. يجب أن تبدأ أسماء المتغيرات بحرف متبوعا بأي عدد من الأرقام أو الأحرف أو underscore. ولا يجوز استخدام الرموز الخاصة أو الفراغ.

٦. جميع أوامر MATLAB تكتب بالحروف الصغيرة (if, while, input, …).

هناك عدة أنواع من المتغيرات في لغة MATLAB وهي:

(أ) المتغيرات العددية Numerical Variables:

تتكون من حرف واحد أو مجموعة من الحروف من A إلى Z و a إلى b ويمكن أن يحتوي على أرقام من 0 إلى 9 ويمكن أن تكون سلسلة من الأرقام والحروف بشرط أن يبدأ بحرف (خليط من أرقام وحروف مبدوءة بحرف) ويمكن كذلك أن يحتوي المتغير على underscore حتى 63 رمزاً. وتكون قيمة المتغير عددية (صحيح، حقيقي، عقدي أو أسي).

مثال:

Ali_Ahmed, X2, S2, ks, K

التعبير الحسابي

يتكون التعبير الحسابي من مجموعة من الثوابت والمتغيرات تجمع بينهما عمليات حسابية ويستخدم فيها الرموز الحسابية مثل +، -، /، *، ^ والأمثلة الآتية تعبر عن تعابير جبرية صيغت بلغة MATLAB.

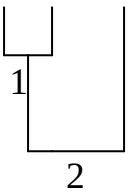
التعبير الجبري	التعبير بلغة MATLAB
$a - 3b$	a – 3 * b
$c^2 - 10$	c ^ 2 – 10
$a^2 + b^2 / 12$	(a ^ 2 + b ^ 2) / 12
$m(7d - 8g)$	m * (7 * d – 8 * g)

قاعدة الأسبقية (الأولوية) Rule of Precedence

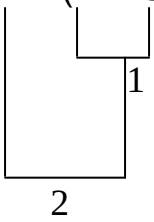
وهذه القاعدة مهمة في فهم وترتيب أولويات العمليات الحسابية في التعابير والمعاملات الحسابية، كما يجريها وينفذها الحاسب، وتنص القاعدة على أن الأولوية الأولى تعطى للعمليات الموجودة بين القوسين ومن اليسار إلى اليمين، وبالنسبة للعمليات الحسابية فالرفع إلى الأس أولا، والضرب (أو القسمة) ثانيا، والجمع (أو الطرح) أخيراً والمثال التالي يوضع هذه القاعدة:

التعبير:

A / B + C يكافئ في الجبر $+ \quad C\frac{A}{B}$



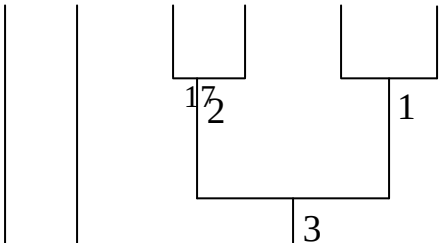
بينما يكافئ التعبير A / (B + C) يكافئ في الجبر $\frac{A}{B+C}$



لان الجمع داخل الأقواس يجري أولاً حسب الأولوية ثم يقسم A على نتيجة القوس.

مثال: التعبير

A - B / (K * F - X ^ M)



تنفيذ العمليات حسب الخطوات التالية:

تأخذ الأقواس الأولوية الأولى، وتنفذ العمليات داخلها حسب الأولوية أيضا.

العملية الأولى: رفع X إلى الأس M لتصبح كمية واحدة.

العملية الثانية: ضرب K في F لتصبح كمية واحدة.

العملية الثالثة: طرح نتيجة العملية الأولى من نتيجة العملية الثانية وتصبح النتيجة كمية واحدة.

العملية الرابعة: تقسم B على نتيجة العملية الثالثة وتصبح النتيجة كمية واحدة.

العملية الخامسة: تطرح نتيجة العملية الرابعة من A وتصبح النتيجة كمية واحدة.

الجملة الحسابية Arithmetic Statement

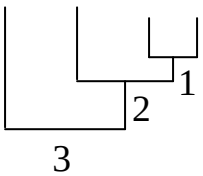
الجملة الحسابية في MATLAB تكافئ المعادلة الحسابية في الجبر إلا أن MATLAB تشترط أن يكون اسم المتغير المراد حساب قيمته في الطرف الأيسر وحده بدون أشارة بينما يكون التعبير الحسابي (بقية المعادلة) في الطرف الأيمن، كما في الأمثلة التالية:

1) y = A * X + B
2) A = 3.14 * R ^ 2

مثال:

أولوية العمليات الحسابية في الجمل الحسابية:

Z = A – B / C



يمكن ملاحظة أن إشارة المساواة تمثل آخر أولوية حسابية بعد انتهاء جميع العمليات الحسابية في الطرف الأيمن.

(ب) المتغيرات الرمزية String Variables:

تشبه في تركيبها المتغيرات العددية والفرق الوحيد بينهما هو أن قيمة المتغير الرمزي تكون رمزية (محصورة بين علامتي اقتباس).

الجملة الرمزية String Statement

تشبه في تركيبها الجملة الحسابية والفرق الوحيد بينهما هو أن المتغير في طرفها الأيمن يكون رمزياً (محصورة بين علامتي اقتباس) والتعبير في طرفها الأيسر يكون متغير.

والأمثلة التالية توضح ذلك:

```
A = 'Hameed Abdul–Kareem';
N = 'Number of Student';
Dept = 'Computer Science';
```

ملاحظة: التعابير في الطرف الأيمن لا يكون لها قيم حسابية لو استخدمت في عمليات حسابية لأنها موضوعة داخل ''.

الاقترانات المكتبية Library Functions:

يتوفر في معظم الحاسبات باستخدام لغة MATLAB اقترانات رياضية يكثر استعمالنا لها، مثل الدوال والاقترانات المثلثية واللوغارتيمية وغيرها ويمكن استدعائها في أي وقت، ومنها:

الاقتران	المعـــــنى
Sqrt	الجذر التربيعي
abs	القيمة المطلقة
exp	المرفوع إلى قوة بأساس 10
log	اللوغاريتم الطبيعي
$\log_{10}$	اللوغاريتم العشري
$\log_{2}$	اللوغاريتم ذو الأساس 2
sin	جيب الزاوية
Cos	جيب تمام الزاوية
Tan	ظل الزاوية
atan	ظل معكوس الزاوية

fix	التدوير باتجاه الصفر
floor	التدوير باتجاه اللانهاية السالبة
ceil	التدوير باتجاه اللانهاية الموجبة
round	التدوير باتجاه أقرب عدد صحيح
mod	الجزء الصحيح من حاصل القسمة
rem	بقية القسمة
Sign	إشارة العدد إذا كانت موجبة، سالبة، صفر
imag	القسم التخيلي
real	القسم الحقيقي
factor	العوامل الأولية
Isprime	يعيد true إذا كان العدد أوليا
primes	ينشئ قائمة بالأعداد الأولية
gcd	القاسم المشترك الأعظم
lcm	المضاعف المشترك الأصغر

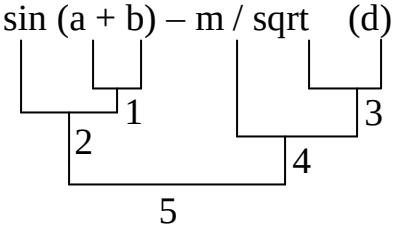
مثال:

```
>> x = 2.6;
>> y1 = fix (x); y2 = floor (x); y3 = ceil (x); y4 = round (x);
y1 = 2
y2 = 2
y3 = 3
y4 = 3
```

س/ مالفرق بين الدوال الأربعة أعلاه؟

ملاحظة:

تأخذ الاقترانات المكتبية أولوية بعد الأقواس عند تنفيذ العمليات الحسابية.



يكون تنفيذ العمليات الحسابية كما يلي:

العملية الأولى: إيجاد قيمة جمع a مع b.

العملية الثانية: إيجاد قيمة جيب الزاوية لناتج العملية (١).

العملية الثالثة: إيجاد قيمة الجذر التربيعي لـ d.

العملية الرابعة: إيجاد ناتج قيمة ناتج قسمة m على ناتج العملية (٣).

العملية الخامسة: طرح ناتج العملية (٤) من ناتج العملية (٢) وتصبح النتيجة النهائية كمية واحدة (عدداً واحداً).

مثال: تمثل الجمل التالية إقترانات مكتبية في الجبر وإزائها قيمتها في MATLAB:

$b = \sqrt{a^2 + 10}$ ⟸ b = sqrt (a ^ 2 + 10)

z = ln (cx + ny) ⟸ z = log (c * x + n * y)

$y = \sin^3 (x + nk)$ ⟸ y = (sin (x + n * k)) ^ 3

$s = \tan^{-1} (y / x)$ ⟸ s = atan (y / x)

$r = 2\sqrt{e^{x-5}}$ ⟸ r = 2 * sqrt (exp (x – 5))

$t = \left|x - \sqrt{y}\right| / (a + m)$ ⟸ t = abs (x – sqrt (y)) / (a + m)

$g = p^{3/2} + \sqrt[5]{ab/c}$ ⟸ g = p ^ (3 / 2) + (a * b / c) ^ (1 / 5)

المصفوفات والعمليات على المصفوفات

لقد كانت جميع الحسابات التي أجريتها حتى الآن مؤلفة من أعداد وحيدة البعد سنسميها أعداد مفردة. وتعتبر العمليات المجراة على الأعداد المفردة هي أساسيات علم الرياضيات. وبنفس الوقت، وعندما يريد الشخص إجراء نفس العملية على عدد مفرد أو أكثر، فسيحتاج إلى أكثر إعادة إجراء العملية عدة مرات، مما يعني هدر في الوقت والجهد. ولحل هذه المشكلة، عمد برنامج MATLAB إلى إجراء العمليات الرياضية على مصفوفة من البيانات.

المصفوفة البسيطة

يتعامل برنامج MATLAB مع المصفوفات بشكل مباشر وبطريقة سلسلة، إذ أن إنشاء المصفوفات يتم بطريقة سهلة جداً.

مثال (١): x = [1, 3, 7, 9, 20]

مثال (٢): y = sin (x) حيث $0 \le x \le \pi$

x	0	0.1π	0.2π	0.3π	0.4π	0.5π	0.6π	0.7π	0.8π	0.9π	π
y	0	0.31	0.59	0.81	0.95	1	0.95	0.81	0.59	0.31	0

```
x = [0  0.1 * pi  .2 * pi  .3 * pi  .4 * pi  .5 * pi  .6 * pi  .7 * pi  .8 * pi  .9 * pi
pi]
y = sin (x)
```

يقتصر كل ما عليك لإنشاء مصفوفة في لغة MATLAB على أن تبدأ بقوس يساري ثم تدخل القيم المطلوبة بفراغ أو (فارزة) ثم أغلق المصفوفة بقوس يميني. وعندما تريد كتابة sin (x) فأن برنامج MATLAB يعلم بأنك تريد حساب الجيب لكل قيم x ويقوم بوضع النتائج في مصفوفة أخرى هي y وتجعل هذه الإمكانية MATLAB مختلفة عن لغات البرمجة الأخرى.

عنونة المصفوفة أو الفهرسة

المصفوفة أعلاه تتكون من ١١ عنصر، يمكن الوصول إلى أي عنصر منها باستخدام الفهرسة له.

```
>> x (3)
ans =
    0.6283
>> y (5)
ans =
    0.9511
```

ولتعريف مجموعة من العناصر بنفس الوقت فأن برنامج MATLAB يستخدم النقطتين المتعامدتين (:).

```
>> x (1: 5)
ans =
    0    0.3142    0.6283    0.9425    1.2566
```

هذه هي العناصر الخمسة الأولى من المصفوفة x، ويجبرك الرمز 1: 5 بأن تبدأ بالرقم 1 وتعدّ حتى الرقم 5.

مثال:

```
>> x (7: end)
ans =
    1.885    2.1991    2.5133    2.8274    3.1416
```

وهنا تكمل من العنصر السابع وحتى نهاية المصفوفة، إذ تشير الكلمة end إلى آخر عنصر من عناصر المصفوفة.

مثال:

```
>> y (3: -1: 1)
```

```
ans =
    0.5878    0.3090      0
```

هنا العنصر الثالث ثم الثاني ثم الأول بترتيب عكسي، ويخبرك الرمز 1 :1- :3 بأن تبدأ بالرقم 3 وتعدّ نزولاً بقيمة 1 وتقف عند الرقم 1.

مثال:

```
>> x (2: 2: 7)
ans =
    0.3142    0.9425    1.5708
```

هنا العنصر الثاني والرابع والسادس من المصفوفة x، ويخبرك الرمز 7 :2 :2 بأن تبدأ بالرقم 2 وتعدّ نحو الأعلى بـ 2 وتقف عندما تصل إلى الرقم 7.

مثال:

```
>> y ([8  2   9   1])
ans =
    0.8090    0.3090    0.5878      0
```

استخدمنا هنا مصفوفة أخرى [1 9 2 8] لوضع عناصر المصفوفة y بالترتيب الذي نرغب فيه، حيث وضع العنصر الثامن أولاً والعنصر الثاني ثانيا، بينما وضع العنصر التاسع ثالثاً والعنصر الأول رابعا. في الواقع تدل المصفوفة [1 9 2 8] عناوين العناصر المرغوبة من المصفوفة y.

مثال:

```
>> y ([1    1    3   4    2    2])
ans =
    0    0    0.5878    0.8090    0.3090    0.3090
```

مثال:

توضح الأمثلة التالية بأن برنامج MATLAB لا يقبل الدليل كرقم غير صحيح حيث يعطي رسالة خطأ.

```
>> y (3.2)
      Error
>> y (3.7)
      Error
```

```
>> y (11.6)
```

خطأ بسبب تجاوز الدليل طول المصفوفة Error

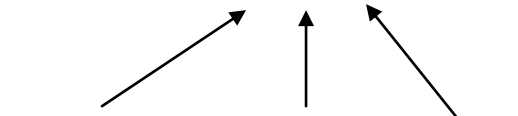
إنشاء المصفوفة

لقد قمنا سابقاً بإدخال قيم مصفوفة x عبر كتابة كل العناصر ضمن المصفوفة، وهنا الأمر مقبول لان المصفوفة x تحوي احد عشر عنصراً فقط، ماذا لو احتوت 111 عنصراً؟

هناك طريقتان لإدخال عناصر المصفوفة x، وذلك باستخدام النقطتين المتعامدتين.

أمثلة:

```
1) >>  x = (0: 0.1: 1) * pi
2) >> x = linspace (0, pi, 11 )
```



مثال:

```
>> a = [1: 7]
a =
   1   2   3   4   5   6   7
```

مثال:

```
>> b = [linspace (1, 7, 5)]
b =
   1   2.5   4   5.5   7
```

مثال:

```
>> a = (1: 7)
a =
   1   2   3   4   5   6   7
```

مثال:

```
>>  a = 1: 5    ,   b = 1: 2: 9
a =
   1   2   3   4   5
```

```
b =
   1   3   5   7   9
```

ملاحظة:

هنا تم إنشاء مصفوفتين، ولكن تذكر بأنك تستطيع دمج التعبيرين ضمن سطر واحد إذا لم تفصل بفواصل:

```
>> c = [b   a]
c =
   1   3   5   7   9   1   2   3   4   5
```

وبذلك تم إنشاء مصفوفة c مؤلفة من عناصر b متبوعة بعناصر a.

تكييف المصفوفة

بالاعتماد على المثال السابق، فان فصل العناصر بفراغات أو بفواصل عادية يحدد عناصر في أعمدة مختلفة، في حين أن استخدام الفاصلة المنقوطة يجعل العناصر واقعة في أسطر مختلفة.

مثال:

```
>> c = [1   2   3   4   5]
c =
   1   2   3   4   5          مصفوفة أفقية
```

مثال:

```
>> c = [1; 2; 3; 4; 5]
c =
   1                    مصفوفة عمودية (كل عنصر في سطر)
   2
   3
   4
   5
```

مثال:

```
>> a = 1: 5
a =
```

1 2 3 4 5

مثال:

```
>> b = a'
```

لقد استخدمنا هنا إشارة المنقول (المدور) لتحويل السطر a إلى العمود b.

```
b =
   1
   2
   3
   4
   5
```

مثال:

```
>> k = b;
```

(تنسيب المصفوفات)

مثال:

```
>> g = [1   2   3   4   ;   5   6   7   8 ]
```

تنزيل سطر آخر

```
g =
   1   2   3   4
   5   6   7   8
```

مصفوفة متكونة من سطرين وأربعة أعمدة

مثال:

```
>> g = [1   2   3  4      ↵
        5   6   7  8      ↵
        9   10  11 12]    ↵
```

كذلك فأن ضغط مفتاح Enter أو Return يخبرنا برنامج MATLAB بأن ينتقل إلى سطر جديد أثناء إدخال قيم المصفوفة.

مثال:

```
>> h = [1   2   3   ;   4   5   6   7]
```

Error عدد الأعمدة غير متساوية

ملاحظة:

عنصر ⟵ half = g (2, 2) ;

مصفوفة ⟵ full = g ;

مثال:

```
>> c = [1: 5; 2: 2: 10; 7: -1: 3]
c =
    1  2  3  4  5
    2  4  6  8  10
    7  6  5  4   3
>> c (1, 2)
 ans =
       2
```

ملاحظة:

تقدم لغة MATLAB طريقة أخرى للإشارة إلى عناصر المصفوفة باستخدام رقم واحد فقط، ولفهم هذه الطريقة يجب التخيل بأن جميع عناصر المصفوفة مرتبة بشكل عمود واحد مكون من أعمدة المصفوفة من الأعلى إلى الأسفل (أي عناصر العمود الأول ثم الثاني ثم الثالث وهكذا).

```
>> c (12)
 ans =
       4
```

العمليات الحسابية بين المصفوفة والعدد المفرد

تجري العديد من العمليات الحسابية كعملية الإضافة والطرح والضرب والقسمة بين العدد المفرد وبين جميع عناصر المصفوفة.

مثال:

```
>> g – 2          % المصفوفة g المعرفة سابقاً
ans =
     -1   0   1   2
      3   4   5   6
```

```
    7   8   9  10
```

وهنا طُرح من كل عنصر من عناصر المصفوفة g العدد 2.

مثال:

```
>> 2 * g – 1
ans =
    1   3   5   7
    9  11  13  15
   17  19  21  23
```

أما هنا فضرب كل عنصر من عناصر المصفوفة g بالعدد 2، ثم طُرح من كل عنصر من العناصر الناتجة الرقم 1.

مثال:

```
>> 2 * g / 5 + 1
ans =
   1.4   1.8   2.2   2.6
     3   3.4   3.8   4.2
   4.6     5   5.4   5.8
```

أما في هذه الحالة، فقد ضُرب كل عنصر من عناصر المصفوفة g بالعدد 2، ثم قُسّم الناتج على العدد 5 وبعدها أُضيف لها الواحد.

العمليات الحسابية بين المصفوفات

لا تعتبر العمليات الحسابية بين المصفوفات بسيطة تماماً مثل العمليات الحسابية المجراة بين المصفوفات والأعداد المفردة. وبشكل أوضح، فالعمليات الحسابية المجراة بين مصفوفات مختلفة الأبعاد والحجوم تعد عمليات صعبة التحديد، وتعد العمليات الحسابية على المصفوفات متساوية الأبعاد من جمع وطرح وضرب وقسمة من العمليات الأساسية في لغة MATLAB واليك الأمثلة التالية:

```
>> g    % إعادة استخدام المصفوفة السابقة
g =
    1   2   3   4
    5   6   7   8
    9  10  11  12
>> h = [1  1  1  1 ; 2  2  2  2 ; 3  3  3  3]
```

```
h =
     1   1   1   1
     2   2   2   2
     3   3   3   3
>> g + h
ans =
      2    3    4    5
      7    8    9   10
     12   13   14   15
>> ans – h
ans =
      1    2    3    4
      5    6    7    8
      9   10   11   12
>> 2 * g – h
ans =
      1    3    5    7
      8   10   12   14
     15   17   19   21
>> 2 * (g – h)
ans =
      0    2    4    6
      6    8   10   12
     12   14   16   18
```

لاحظ أيضاً بأن العمليات الحسابية بين المصفوفات تعتمد نفس تسلسل أسبقية العمليات المعتمد عند إجراء العمليات الحسابية على الأعداد المفردة، ويمكن أيضاً استخدام الأقواس لكسر تلك الأولوية. كما

ويمكن ضرب كل عنصر بالعنصر المناظر لـه من المصفوفة الأخرى أو قسمته شرط إن تُسبق إشارة الضرب أو القسمة بنقطة كما في الشكل:

```
>> g .* h
ans =
     1     2     3     4
    10    12    14    16
    27    30    33    36
```

ولقد قمنا هنا بضرب المصفوفة g بالمصفوفة h عنصر بعنصر عبر استخدام إشارة الضرب المسبوقة بنقطة.

ملاحظة: يجعل وجود النقطة أمام إشارة الضرب القياسية برنامج MATLAB يضرب المصفوفتين عنصرا بعنصر، بينما تخبر إشارة الضرب لوحدها البرنامج بان يقوم بضرب مصفوفات عادية.

```
>> g * h
Error
```

لان عدد الأسطر للمصفوفة g $\neq$ عدد الأعمدة للمصفوفة h

كما إن قسمة مصفوفتين عنصراً بعنصر ممكنة عن طريق كتابة إشارة القسمة مسبوقة بنقطة كما في المثال التالي:

```
>> g ./ h
ans =
    1.0000    2.0000    3.0000    4.0000
    2.5000    3.0000    3.5000    4.0000
    3.0000    3.3333    3.6667    4.0000
```

ملاحظة: إذا سبقت إحدى إشارة القسمة بنقطة، عندها سيقوم برنامج MATLAB بتقسيم المصفوفتين عنصرا بعنصر. أما إذا كانت القسمة بدون نقطة، فإننا ستحدد قسمة مصفوفات عادية.

```
>> g .^ 2
ans =
     1     4     9    16
    25    36    49    64
```

81 100 121 144

ولقد وجدنا هنا مربع كل عنصر من عناصر المصفوفة g.

المصفوفات القياسية

يمكّنك برنامج MATLAB من إنشاء مصفوفات قياسية، وذلك لتمتع تلك المصفوفات بخواص وميزات خاصة، وتتضمن أيضاً المصفوفات التي جميع عناصرها صفرية أو مساوية للواحد، ومصفوفات الأعداد العشوائية والمصفوفات القطرية والمصفوفات التي عناصرها أعداد ثابتة.

```
>> ones (3)          (مصفوفة واحديه)
ans =
     1   1   1
     1   1   1
     1   1   1
>> zeros (2, 5)       (مصفوفة صفرية)
        ↑     ↑
      الأسطر  الأعمدة
ans =
     0   0   0   0
     0   0   0   0
>> size (g)           (تحديد أبعاد مصفوفة)
ans =
      3   4
>> ones (size (g))
ans =
     1   1   1   1
     1   1   1   1
     1   1   1   1
```

ملاحظة: عندما يتبع اسم المصفوفة القياسية برقم مفرد مثل ones (n) أو zeros (n) فأن برنامج MATLAB ينشىء مصفوفات مربعة n×n تحتوي على أصفارا أو واحديه على الترتيب.

```
>> eye (4)            (مصفوفة الوحدة)
```

```
ans =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
>> rand (3)          (مصفوفة عشوائية)
ans =
    0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214
>> rand (1, 5)
ans =
    0.4447    0.6154    0.7919    0.9218    0.7382
```

مثال:

```
>> d = pi;
>> d * ones (3, 4)
ans =
    3.1416    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416    3.1416
>> d + zeros (3, 4)
ans =
    3.1416    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416    3.1416
>> repmat (d, 3, 4)        (تكرار القيمة d بالأبعاد 3×4)
ans =
```

```
3.1416    3.1416    3.1416    3.1416
3.1416    3.1416    3.1416    3.1416
3.1416    3.1416    3.1416    3.1416
```

ملاحظة: يمكن ان تكون d مصفوفة فتكون حينئذ تكرر مصفوفات وليس قيم.

التعامل مع المصفوفة

لقد امتلك برنامج MATLAB العديد من الطرق للتعامل مع المصفوفات، وكانت هذه الخاصية هي أهم مميزات البرنامج، فما إن تُحدّد المصفوفة حتى يزودك البرنامج بأقوى طرق الإدخال، التوسعة أو إعادة ترتيب بعض أجزاء المصفوفة عبر استعمال تعابير أو تعليمات محددة وممتعة، وتعتبر معرفة هذه التعليمات مفتاح الاستعمال الفعال لبرنامج MATLAB. ولشرح التعامل مع المصفوفات نأخذ الأمثلة التالية:

```
>> A = [1  2  3; 4  5  6; 7  8  9]
A =
    1    2    3
    4    5    6
    7    8    9
>> A (3, 3) = 0
A =
    1    2    3
    4    5    6
    7    8    0
```

جعل العنصر في الموقع (3, 3) صفراً.

```
>> A (2, 6) = 1
A =
    1    2    3    0    0    0
    4    5    6    0    0    1
    7    8    0    0    0    0
>> A (:, 4) = 4
A =
```

جعل العنصر في الموقع (2, 6) تكون 1 وبما ان المصفوفة A لا تمتلك ستة أعمدة، لذلك سيقوم البرنامج بتوسيعها حسب الضرورة ويضع بقي العناصر صفراً وتكون مستطيلة.

```
    1   2   3   4   0   0
    4   5   6   4   0   1
    7   8   0   4   0   0
>> A (:, 4) = [4; 4; 4]
A =
    1   2   3   4   0   0
    4   5   6   4   0   1
    7   8   0   4   0   0
>> A (:, 4) = [4  4  4]
```

جعل جميع عناصر العمود الرابع تكون 4

جعل جميع عناصر العمود الرابع تكون 4

Error بسبب عدم وجود فارزة منقوطة

مثال:

```
>> B = [7   8   9;  4   5   6; 1   2   3]
B =
    7   8   9
    4   5   6
    1   2   3
>> A = [1   2   3; 4   5   6; 7   8   9]
A =
    1   2   3
    4   5   6
    7   8   9
>> C = [A   B(:, [1  3])]
C =
    1   2   3   7   9
    4   5   6   4   6
    7   8   9   1   3
>> B = A (1: 2, 2: 3)
```

حصلنا على المصفوفة C عبر توسيع المصفوفة A بإضافة العمودين الأول والثالث من المصفوفة B.

```
B =
    2   3
    5   6
```

مثال: تشكيل المصفوفة B بجعل المصفوفة A كمصفوفة عمود واخذ أعمدتها عمود بعد عمود.

```
>> B = A (:)
B =
    1
    4
    7
    2
    5
    8
    3
    6
    9
```

مثال:

```
>> B = [1  2  3; 4  5  6; 7  8  9];
>> B = reshape (A, 1, 9)
B =
    1   4   7   2   5   8   3   6   9
>> B = reshape (A, [1  9]);
B =
   1   4   7   2   5   8   3   6   9
```

في المثال أعلاه إيعاز تحويل أبعاد المصفوفة الثنائية 3*3 إلى مصفوفة أحادية 9*1

مثال:

```
>> A = B
A =
```

```
    1   2   3
    4   5   6
    7   8   9
>> B (:, 2) = [ ]
B =
    1   3
    4   6
    7   9
```

تمت إعادة صياغة المصفوفة B عبر حذف كل اسطر العمود الثاني من المصفوفة B الأصلية، وعندما تضع أي عنصر مساوياً للمصفوفة الفارغة [ ]، فهذا يعني انك تريد حذفها من المصفوفة وتقليصها لتحافظ على العناصر المتبقية بعد الحذف.

مثال: إيجاد منقول (مدور) المصفوفة وإعادة تشكيلها بالتعليمة reshape.

```
>> C = B'
C =
    1   4   7
    3   6   9
>> reshape (B, 2, 3)
ans =
     1   7   6
     4   3   9
```

ملاحظة: تعمل تعليمة reshape عمود بعد عمود وللحصول على سطر بعد سطر نعمل المدور (transport).

مثال: هنا حذفنا السطر الثاني في المصفوفة C.

```
 >> C (2, :) = [ ]
C =
    1   4   7
```

مثال: استبدلنا عناصر السطر الثاني من المصفوفة A بعناصر C.

```
>> A (2, :) = C
A =
     1   2   3
```

```
    1   4   7
    7   8   9
```

مثال:

```
>> x = -3: 3
x =
   -3   -2   -1   0   1   2  3
```

هناك المصفوفات المنطقية الناتجة عن العمليات المنطقية. كما يمكن أيضاً استخدام المصفوفات المنطقية إذا كان حجمها مساويا لحجم المصفوفات المعنونة، ويتم في هذه الحالة الإبقاء على العناصر ذات القيمة (1) أي true وهي العناصر المحققة للشرط بينما يتجاهل العناصر (0) أي false وهي العناصر غير المحققة الشرط. ولنأخذ المثال التالي:

```
>> abs (x) > 1
ans =
    1   1   0   0   0   1   1
>> y = x (abs (x) > 1)
```

هنا تم إنشاء المصفوفة y من تلك العناصر من المصفوفة x التي قيمتها اكبر من الواحد.

```
y =
   -3   -2   2   3
```

ويمكن العمل مع المصفوفات الثنائية المنطقية كما عملنا مع الأحادية المنطقية، كما في المثال التالي:

```
>> B = [5   -3; 2   -4]
B =
   5   -3
   2   -4
>> x = abs (B) > 2
x =

   1   1
   0   1
```

```
>>  y = B (x)
y =
    5
    -3
    -4
```

ترتيب المصفوفة

عندما تعطى متجه من البيانات فان أهم عملية يمكن إن نحتاجها وتود تطبيقها هي الترتيب، ويؤمن الايعاز sort عملية الترتيب في لغة MATLAB، كما هو واضح في المثال التالي:

```
>> x = randperm (8)                    (إيعاز ترتيب الأرقام بصورة عشوائية)
x =
    7   5   2   1   3   6   4   8
>> [y, indx] = sort (x)
```

الترتيب الجديد المواقع القديمة

```
y =
    1   2   3   4   5   6   7   8
indx =
    4   3   5   7   2   6   1   8
```

وعندما تكون المصفوفة ثنائية البعد فان عملية الترتيب تتم بشكل مختلف وكما يلي:

(عمود بعد عمود)

```
>> A = [randperm (6); randperm (6); randperm (6); randperm (6)]
A =
    1   2   5   6   4   3
    4   2   6   5   3   3
```

```
    2   3   6   1   4   5
    3   5   1   2   4   6
>> [As, idx] = sort (A)
As =
    1   2   1   1   3   1
    2   2   5   2   4   3
    3   3   6   5   4   5
    4   5   6   6   4   6
idx =
    1   1   4   3   2   2
    3   2   1   4   1   1
    4   3   2   2   3   3
    2   4   3   1   4   4
```

البحث عن مصفوفة جزئية

من المفيد في بعض الأحيان إن تعرف موقع أو دليل العناصر التي تحقق شرطا معينا، والموجودة ضمن مصفوفة معينة. يقوم برنامج MATLAB بتحقيق هذه الغاية عبر الايعاز find، والذي يعيد لك دليل أو موقع العنصر الذي تكون نتيجة تحقيقه لشرط ما true، واليك المثال التالي:

```
>> x = -3: 3
x =
    -3   -2   -1   0   1   2   3
>> k = find (abs (x) > 1)
 k =              (الموقع)
    1   2   6   7
>> y = x (k)
y =
    -3   -2   2   3
>> y = x (abs (x) > 1)
```

```
y =
    -3    -2     2     3
```

ويستطيع الايعاز find أن يعمل في المصفوفات الثنائية البعد أيضا (عمود بعد عمود)، فمثلا:

```
>> A = [1    2    3;    4    5    6;    7    8    9]
A =
    1    2    3
    4    5    6
    7    8    9
>> [i, j] = find (A > 6)
i =
    3
    3
    3
j =
    1
    2
    3
```

ملاحظة: الايعاز diag يوجد عناصر القطر الرئيسي للمصفوفة.

$$A = \begin{bmatrix} \mathbf{7} & 8 & 9 & 9 \\ 7 & \mathbf{8} & 9 & 9 \\ 4 & 5 & \mathbf{8} & 6 \\ 7 & 8 & 9 & \mathbf{9} \end{bmatrix}$$

```
>> diag (A)
ans =
     7
     8
     8
```

9

ملاحظة:

يوفر برنامج MATLAB الدالتين max ، min الذين يوجدان اكبر واصغر عنصر في المصفوفة ومواقعهما.

في حالة المصفوفة الأحادية:

```
>> v = rand (1, 6)
v =
   0.3046   0.1897   0.1934   0.6822   0.3028   0.5417
>> max (v)
ans =
      0.6822
>> [mx, i] = max (v)
mx =
      0.6822
i =
   4
>> min (v)
ans =
      0.1897
>> [mn, j] = min (v)
mn =
      0.1897
j =
   2
```

في حالة كون المصفوفة ثنائية البعد:

```
>> A = rand (4, 6)
A =
```

```
    0.1509    0.8537    0.8216    0.3420    0.7271    0.3704
    0.6979    0.5936    0.6449    0.2897    0.3093    0.7027
    0.3784    0.4966    0.8180    0.3412    0.8385    0.5466
    0.8600    0.8998    0.6602    0.5341    0.5681    0.4449
>> [mx, r] = max (A)
mx =
    0.8600    0.8998    0.8216    0.5341    0.8385    0.7027
r =
    4    4    1    4    3    2
```

ملاحظة:

```
>> max (A');                (اكبر عنصر لكل سطر)
>> [mn, r] = min (A)
mn =
    0.1509    0.4966    0.6449    0.2897    0.3093    0.3704
r =
    1    3    2    2    2    1
```

ملاحظة:

```
>> min (A');                (اصغر عنصر لكل سطر)
```

<<ملاحظة: اكبر عنصر في مصفوفة ثنائية البعد.

```
mmx = max (mx)
mmx =
    0.8998
>> [mmx, i] = max (A (:))
mmx =
    0.8998
i =
    8
```

ملاحظة: توجد طريقة أخرى:

```
>> z = max (max (A));
>> z = min (min (A));
```

ملاحظة: نفس الشيء لحساب المجموع sum.

```
>> z = sum (sum (A));
```

توابع التعامل مع المصفوفة

يزودك برنامج MATLAB، بالإضافة إلى عنونة المصفوفات والمقدرة على التعامل مع المصفوفات التي شرحناها سابقاً، بعمليات التعامل مع المصفوفات، وهي سهلة التطبيق مثل:

```
>> A = [1   2   3; 4   5   6; 7   8   9]
A =
    1   2   3
    4   5   6
    7   8   9
>> flipud (A)                قلب المصفوفة باتجاه up-down
ans =
      7   8   9
      4   5   6
      1   2   3
>> fliplr (A)                قلب المصفوفة باتجاه left-right
ans =
      3   2   1
      6   5   4
      9   8   7
>> triu (A)                  استخلاص الجزء المثلية العليا (upper)
ans =
      1   2   3
      0   5   6
```

```
     0   0   9
>> tril (A)          (lower) استخلاص الجزء المثلية السفلى
ans =
     1   0   0
     4   5   0
     7   8   9
>> g = det (A);        حساب محدد المصفوفة (قيمة)
>> h = inv (A);       حساب معكوس المصفوفة (مصفوفة)
>> i = eig (A);       حساب القيم الذاتية للمصفوفة
>> j = eye (3)            حساب مصفوفة الوحدة
j =
   1   0   0
   0   1   0
   0   0   1
>> trace (A);         حساب مجموع عناصر القطر الرئيسي
```

حجم المصفوفة

إذا أردت أن تعرف حجم أو بعد مصفوفة أحادية أو ثنائية أو ثلاثية البعد غير معروفين وكنت بحاجة لحجمها لإجراء بعض العمليات الرياضية، فان برنامج MATLAB يمكنك من خلال الايعاز length و size و numel واليك الأمثلة التالية:

```
>> A = [1   2   3   4; 5   6   7   8]
A =
   1   2   3   4
   5   6   7   8
>> S = size (A)
S =
   2   4
```

يعبر العنصر الأول عن عدد الأسطر (2) بينما يعطي العنصر الثاني عدد الأعمدة (4).

```
>> [r, c] = size (A)
r =
    2
c =
    4
>> r = size (A, 1)
r =
    2
>> c = size (A, 2)
c =
    4
```

يعيد الايعاز numel العدد الكلي لعناصر مصفوفة فمثلاً:

```
>> numel (A)
ans =
      8
```

بينما يعيد الايعاز length عدد العناصر الموجودة ضمن البعد الأطول للمصفوفة، كما يلي:

```
>> length (A)
ans =
      4
>> B = -3: 3
B =
    -3    -2    -1    0    1    2    3
>> length (B)
ans =
     7
>> min (size (A))          A مصفوفة ثنائية
ans =
```

2

ملاحظة: طريقة توليد مصفوفة بالدمج.

```
>> x = [1    2; 3    4];
>> y = [x    x .^ 2; x .^ 3    x .^ 4];
```

المصفوفات متعددة الإبعاد

لقد شرحنا في الفصل السابق المصفوفات أحادية وثنائية الأبعاد والعمليات التي تجري عليها. يدعم برنامج MATLAB المصفوفات متعددة الأبعاد (أي n-D arrays) وذلك نفس الايعازات وتقنيات العنونة المطبقة على المصفوفات أحادية وثنائية البعد. وبشكل عام، يرقم البعد الثالث عبر صفحات (pages)، ولذلك تمتلك المصفوفات ثلاثية البعد اسطرا وأعمدة وصفحات، حيث تتألف كل صفحة من

مصفوفة ثنائية البعد ذات اسطر وأعمدة، ويجب أن تمتلك كل صفحة عددا متساويا من الأسطر والأعمدة والعكس بالعكس في كل صفحة.

ليس هناك حد لعدد الأبعاد في المصفوفات، ولكننا سنستخدم مصفوفات ثلاثية الأبعاد في هذا الفصل بسبب سهولة تخيلها وإظهارها.

تركيب المصفوفة

يمكن إنشاء المصفوفة المتعددة الأبعاد بطرق مختلفة، واليك بعضها:

```
>> A = zeros (4, 3, 2)
A (:, :, 1) =
     0   0   0
     0   0   0
     0   0   0
     0   0   0
A (:, :, 2) =
     0   0   0
     0   0   0
     0   0   0
     0   0   0
```

تتألف هذه المصفوفة الصفرية من أربعة اسطر وثلاثة أعمدة وصفحتين، ولقد ظهرت الصفحة الأولى ثم الصفحة الثانية.

مثال:

```
>> B (:, :, 1) = zeros (2, 3);
>> B (:, :, 2) = ones (2, 3);
>> B (:, :, 3) = 4;
>> B
B (:, :, 1) =
     0   0   0
     0   0   0
```

```
B (:, :, 2) =
        1   1   1
        1   1   1
B (:, :, 3) =
        4   4   4
        4   4   4
```

يمكن استخدام الايعاز reshape لتحويل المصفوفة من مصفوفة ثنائية الأبعاد إلى مصفوفة ثلاثية الأبعاد وكالاتي:

```
>> C = [B (:, :, 1), B (:, :, 2), B (:, :, 3)]
C =
    0   0   0   1   1   1   4   4   4
    0   0   0   1   1   1   4   4   4
>> reshape (C, 2, 3, 3)
ans (:, :, 1) =
        0   0   0
        0   0   0
ans (:, :, 2) =
        1   1   1
        1   1   1
ans (:, :, 3) =
        4   4   4
        4   4   4
```

حجم المصفوفة

الايعاز size يعيد بعد المصفوفة وفق كل أبعادها كما شرحنا سابقاً.

```
>> [r, c, p] = size (C)
r =
    2
```

```
c =
     3
p =
     3
```

وإذا لم نعرف عدد إبعاد المصفوفة أو كانت ابعادها متغيرة، عندما نستطيع استخدام الايعاز ndims كما يلي:

```
>> ndims (C)
ans =
      3
>> numel (C)                إيجاد عدد عناصر المصفوفة
ans =
      18
>> length (size (C))        إيجاد طول اكبر بعد بالمصفوفة
ans =
      3
```

مصفوفة الخلايا Cell Arrays

تعتبر مصفوفات الخلايا مصفوفات في لغة MATLAB تكون عناصرها عبارة عن خلايا، وتتضمن كل خلية نوعاً من البيانات قد تكون مصفوفات عددية أو رمزية أو كائنات بسيطة أو مصفوفات خلايا أخرى. فمثلاً قد تحوي خلية من مصفوفة الخلية مصفوفة عددية وتحوي الخلية الأخرى مصفوفة رمزية، بينما تحوي الثالثة على أعداد عقدية (يسمح باستخدام مصفوفات بأنواع بيانية مختلفة (غير متجانسة))

كما ويمكن إنشاء مصفوفات الخلايا بأي بعد كان كما هي الحال مع المصفوفات العددية، ولكن معظم مصفوفات الخلايا تكون عبارة عن مصفوفات أحادية البعد.

تنشأ مصفوفات الخلايا عبر استخدام تعابير الإسناد أو عبر إعادة تقسيم المصفوفة بالإيعاز cell، ثم نقوم بإسناد البيانات إلى الخلايا.

هناك طريقتان مختلفتان للوصول إلى الخلايا. وإذا أردت استخدام رموز مصفوفة قياسية للدلالة على المصفوفة، يجب عليك أن تحيط الخلية بأقواس مجموعة { }، إذ إن برنامج MATLAB يستخدم هذه الأقواس لتعريف مصفوفات الخلايا، واليك الأمثلة التالية:

```
>> A (1, 1) = {[1   2   3; 4   5   6; 7   8   9]};
>> A (1, 2) = {2 + 3i};
>> A (2, 1) = {'Ali Ahmed'};
>> A (2, 2) = {12: -2: 0};
>> A
A =
    [3×3 double]        [2.0000 + 3.0000i]
    'Ali Ahmed'         [1×7 double]
```

لاحظ إن برنامج MATLAB يظهر المصفوفة كمصفوفة خلية بعداها 2×2 ولكن ذلك لا يظهر محتويات الخلية، وإنما يظهر البرنامج محتويات الخلية بشكل أساسي إذا لم تأخذ هذه المحتويات حجماً كبيراً، كما ويوصف محتويات الخلية إذا أخذت حجماً معقولاً. إن وجود أقواس مجموعات على الجانب الأيمن من المساواة يدل على إن المشار إليه هو خلية وليس قيما عددية وهذا ما يسمى *بفهرسة الخلية (cell indexing)*، وسينشئ التعابير التالية مصفوفة الخلية نفسها.

ملاحظة:

يخبر كلا التعبيرين A (i, j) = {x} و A {i, j} = x برنامج MATLAB بأن يضع المتغير x في العنصر (i, j) من مصفوفة الخلية A.

يدعى التعبير A (i, j) *بفهرسة الخلية (cell indexing)*، بينما يدعى التعبير A {i, j} *بعنونة المحتوى (content addressing)* أي تدل أقواس المجموعة { } على محتوى الخلية، بينما تعرف الأقواس العادية ( ) الخلايا دون النظر إلى محتواها.

مثال:

```
>> y = {1, 'hello', 1 > 5}
y =
    [1]    'hello'    [0]
>> y {1}
ans =
       1
>> y {2}
ans =
       hello
>> y {3}
ans =
       0
```

مثال:

```
>> ce = {[1    2    3; 5    6    7], 'yes', 3 > 2};
>> ce {1}(2, 2)
ans =
       6
```

مثال:

```
>> x = rand (3, 3);
>> y = rand (3, 3);
>> z = rand (3, 3);
>> w {1} = x;
>> w {2} = y;
>> w {3} = z;
>> w
ans =
```

[3×3 double] [3×3 double] [3×3 double]

مثال:

```
>> x {1} = rand (3, 3);
>> x {2} = rand (3, 3);
>> x {3} = rand (3, 3);
      .
      .
      .
      .
      .
>> x {9} = rand (3, 3);
>> x {1}
ans =
    0.8462   0.6721   0.6813
    0.5252   0.8381   0.3795
    0.2026   0.0196   0.8318
```

>> x {1} (2, 2) العنصر الموجود في السطر الثاني والعمود الثاني في مصفوفة (الخلية) الأولى

```
ans =
    0.8381
```

مثال: برنامج لجمع المصفوفات التسعة في المثال السابق في مصفوفة واحدة.

```
>> L = length (x);
>> sum1 = 0;
>> for i = 1: L
     b = x {i};
     sum1 = sum1 + b;
   end;
```

يجبر الإيعاز celldisp برنامج MATLAB على إظهار محتوى الخلايا بالنموذج العادي، واليك المثال التالي الذي يوضح ذلك:

```
>> celldisp (A)
A (1, 1) =
     1   2   3
     4   5   6
     7   8   9
  A (2, 1) =
      Ali Ahmed
  A (1, 2) =
      2.0000 + 3.0000i
  A (2, 2) =
      12   10   8   6   4   2   0
```

كما يُظهر البرنامج محتوى الخلية المفردة عبر طلب محتوى الخلية باستخدام عنونة المحتوى، وهذا يتم بشكل مختلف عن فهرسة الخلية التي تعّرف الخلية فقط دون الدخول إلى محتوى الخلية، فمثلا:

```
>> A {2, 2}
ans =
    12   10   8   6   4   2   0
>> A (2, 2)
ans =
    [1×7 double]
>> A (1, :)
ans =
    [3×3 double]    [2.0000 + 3.0000i]
```

لاحظ بأن البرنامج استخدم لجميع الخلايا السابقة الاسم ans، وذلك لان خلايا البيانات المخزونة ليس لها اسم محدد.

لقد استخدمنا سابقاً الأقواس المربعة لإنشاء المصفوفات العددية، وتعمل أقواس المجموعة نفس العمل بالنسبة للخلايا، وتفصل الأعمدة بفواصل بينما تفصل الأسطر بفواصل منقوطة. واليك المثال التالي:

```
>> B = {[1     2], 'John Smith';  2 + 3i, 5}
B =
    [1×2 double]          'John Smith'
    [2.0000 + 3.0000i]    [5]
```

من المألوف عند استخدام المصفوفات العددية أن تُملأ المصفوفة بعناصر صفرية ثم تُملأ من جديد بالبيانات اللازمة، ويمكن استخدام نفس المنهج في مصفوفات الخلايا، حيث ينشأ الايعاز cell مصفوفة خلية ويملؤها بمصفوفات عددية فارغة [ ] ولنأخذ المثال التالي:

```
>> C = cell (2, 3)
C =
    [ ]   [ ]   [ ]
    [ ]   [ ]   [ ]
```

ما إن يتم تعريف مصفوفة الخلية فأنة يمكن تعميم الخلايا عن طريق عنونة المحتوى وفهرسة الخلايا، كما يبينه المثال التالي:

```
>> C (1, 1) = 'The does n't work'
    Error
```

لقد استخدمنا هنا في الجانب الأيسر دليل الخلية وبالتالي، يجب أن يكون الطرف الأيمن خلية وهذا ما سبب ظهور الخطأ، وليس كوننا لم نُخط محتوياتها بأقواس مجموعة.

```
>> C (1, 1) = {'The does n't work'}
C =
    'The does n't work'   [ ]   [ ]
    [ ]                   [ ]   [ ]
>> C (2, 3) = {'This works too'}
C =
    'This does work'      [ ]          [ ]
    [ ]                   [ ]          'This works too'
```

وبسبب وجود أقواس المجموعة في الجانب الأيسر من العبارة الأخيرة، فان برنامج MATLAB سيضع الخيط الرمزي في الخلية المعنونة. ويوجد هنا مرة أخرى عنونة محتوى، بينما تعتبر العبارة الأصلية مثالاً عن فهرسة المصفوفة.

التعامل مع مصفوفة الخلية

يمكن أن نستخدم الأقواس المربعة أيضاً لضم مصفوفات الخلايا ضمن مصفوفات اكبر، كما هو الحال للمصفوفة العددية، واليك المثال التالي:

```
>> A
A=
    [3×3 double]          [2.0000 + 3.0000i]
    'Ali Ahmed'           [1×7 double]

>> B
B =
     [1×2 double]          'John Smith'
     [2.0000 + 3.0000i]    [5]
>> C = [A; B]            (متساوية الأبعاد)
C =
    [3×3 double]          [2.0000 + 3.0000i]
    'Ali Ahmed'           [1×7 double]
    [1×2 double]          'John Smith'
    [2.0000 + 3.0000i]    [5]
```

يمكن تحديد خلايا جزئية لإنشاء خلايا جديدة عبر استخدام تقنيات مناسبة لعنونة مصفوفة الخلية كما في المثال التالي:

```
>> D = C ([1 3], :)
D =
     [3×3 double]          [2.0000 + 3.0000i]
     [1×2 double]          'John Smith'
```

ويمكن حذف سطر مصفوفة الخلية عبر استخدام الخلية الفارغة.

```
>> C (3, :) = [ ]
C =
    [3×3 double]              [2.0000 + 3.0000i]
    'Ali Ahmed'               [1×7 double]
    [2.0000 + 3.0000i]        [5]
```

ويستخدم الايعاز reshape لتغير مواضع الخلايا، ولكنه لا يستطيع إضافة أو حذف الخلايا وليبان ذلك، نأخذ المثال التالي:

```
>> x = cells (3, 4);
>> size (x)
ans =
      3    4
>> y = reshape (x, 6, 2);
>> size (y)
ans =
      6     2
```

أي إن الايعاز reshape يعيد تشكيل أية مصفوفة بدون تغير نوعها، وكذلك يعيد الايعاز size حجم أي نوع من المصفوفات.

كذلك يعيد الايعاز repmat بالتعامل مع مصفوفات الخلايا حيث يعمل على تكرارها وفقاً للتكرار المطلوب.

<u>مثال:</u>

```
>> y
y =
    [ ]    [ ]
    [ ]    [ ]
    [ ]    [ ]
    [ ]    [ ]
```

```
    [ ]   [ ]
    [ ]   [ ]
>> z = repmat (y, 1, 3)
z =
    [ ]   [ ]   [ ]   [ ]   [ ]   [ ]
    [ ]   [ ]   [ ]   [ ]   [ ]   [ ]
    [ ]   [ ]   [ ]   [ ]   [ ]   [ ]
    [ ]   [ ]   [ ]   [ ]   [ ]   [ ]
    [ ]   [ ]   [ ]   [ ]   [ ]   [ ]
    [ ]   [ ]   [ ]   [ ]   [ ]   [ ]
```

السلاسل الرمزية

تكمن قوة برنامج MATLAB الحقيقية في القدرة على التعامل مع الأرقام، ولكنه يحتاج أحياناً إلى التعامل مع النصوص كما في حالة وضع العناوين وأسماء المحاور على الرسومات.

تركيب السلسة الرمزية

تتألف السلاسل الرمزية في لغة MATLAB من مصفوفات عددية خاصة من قيم ASCII والتي تعيد أظهار السلسة الرمزية، فمثلاً:

```
>> t = 'How about this character string?'
t =
    How about this character string?
```

```
>> size (t)
ans =
     1    32
>> whos                    إيعاز عرض أسماء المتغيرات وحجومها وعدد بياناتها وصنفها
```

Name	Size	Bytes	Class
ans	1×2	16	double array
t	1×32	64	character array

Grand total is 34 elements using 80 bytes

إن السلاسل الرمزية ببساطة هي نص محاطة بفاصلة علوية مفردة. ويعتبر كل حرف في السلسة عنصراً من مصفوفة، والتي نحتاج إلى بايتين لتخزين كل حرف، ونختلف بذلك عن 8 بايت المخصصة لكل عنصر من عناصر المصفوفة العددية أو مضاعفة الدقة. ولرؤية التمثيل ASCII لسلسلة رمزية نحتاج فقط لإجراء بعض العمليات الرياضية على السلسلة أو استخدام الايعاز double، وكما في المثال التالي:

```
>> u = double (t)
u =
  Columns 1 through 12
    72   111   119    32    97    98   111   117   116    32   116   104
  Columns 13 through 24
   105   115    32    99   104    97   114    97    99   116   101   114
  Columns 25 through 32
    32   115   116   114   105   110   103    63
>> char (u)
ans =
     How about this character string?
>> char (u (1))
ans =
     H
```

وبما إن السلاسل هي مصفوفات، لذلك يمكن التعامل معها بكل أدوات التعامل مع المصفوفات المتوفرة في لغة MATLAB، فمثلاً:

```
>>  u = t (16: 24)
u =
    character
```

وتعنون السلاسل تماماً كما تعنون المصفوفات، وتحوي العناصر من 16 إلى 24 في المثال أعلاه الكلمة character

```
>> u = t (24: -1: 16)
u =
    retcarahc
```

وهنا تمت تهجئة الكلمة character بشكل عكسي.

```
>> u = t (16: 24)'
u =
     c
     h

       a
       r
       a
       c
       t
       e
       r
```

وتم هنا تحويل كلمة character إلى مصفوفة عمود عبر عملية مدور(منقول).

يمكن دمج المصفوفات الرمزية وكالاتي:

```
>> u = 'Hameed ';
>> v = 'Aiad';
```

```
>> w  = [u  v]
w =
    Hameed Aiad
```

ويسمح لنا الايعاز disp إظهار السلسلة بدون طباعة اسم المتغير كما في المثال التالي:

```
>> disp (u)
   Hameed
```

ويمكن أن تتملك السلاسل الرمزية (كما في باقي المصفوفات) عدة اسطر، ولكن يجب أن يحوي كل سطر عدداً متساوياً من الأعمدة، لذلك يجب استخدام الفراغات لجعل طول كل الأسطر متساوية كما في المثال التالي:

```
>> v = ['character strings having more than   '
        'one row must have the same number'
        'of columns just like arrays!            ']
v =
    character string having more than
    one row must have the same number
    of columns just like array!
```

وينشئ الايعاز char مصفوفة نصية متعددة الأسطر انطلاقاً من سلاسل مستقلة مختلفة الطول، كما في المثال التالي:

```
>> legends = char ('Wilt', 'Russel', 'Kareem', 'Bird', 'Magic', 'Jordan')
legends =
          Wilt
           Russel
           Kareem
           Bird
          Magic
          Jordan
>> size (legends)
```

```
ans =
     6     6
```

تحويل الأعداد إلى سلاسل رمزية وبالعكس

قد نرغب في العديد من الحالات بتحويل النتائج العددية إلى سلاسل رمزية واستخراج البيانات العددية من السلاسل الرمزية. يزودك برنامج MATLAB بالإيعاز num2str و int2str و fprintf وغيرها لتحويل النتائج العددية إلى سلاسل رمزية، واليك الأمثلة التالية على التحويل:

```
>> int2str (35)
ans =
     35
>> class (ans)
 ans =
      char
>> num2str (3.5)
ans =
     3.5
>> class (ans)
ans =
     char
>> fprintf ('% 4.3f\n', sqrt (2))
  1.414
>> size (fprintf ('% 4.3f\n' , sqrt (2))
ans =
     1     1
```

مثال:

```
>> radius = sqrt (2);
>> area = pi * radius ^ 2;
>> fprintf ('A circle of radius% 6.4f has an area of % 6.4f', radius, area)
```

A circle of radius 1.4142 has an area of 6.2832

تحدد هنا الصيغة 6.4f % ست خانات لإظهار المتغير radius والمتغير area.

مثال (طريقة أخرى):

```
>> S = ['A circle of radius ', (num2str (radius)), 'has an of ',  (num2str (area))  '.']
S =
A circle of radius 1.4121 has an area of 6.2832.
```

تحويل السلاسل الرمزية إلى عددية

```
>> S = str2num ('3.5')
S =
     3.5
>> t = ['3.5▼'      'sqrt(2)'   ;'▼1.5'  '▼▼▼▼9.5']
```

فراغات ▼

يجب إن تكون أطوال الأسطر متساوية

```
t =
   3.5 sqrt(2)
    1.5    9.5
```

← مصفوفة رمزية

```
>> str2num (t)
ans =
      3.5000    1.4142
      1.5000    9.5000
```

← مصفوفة عددية

```
>> t = '[3.5▼sqrt(2);▼1.5▼9.5]'
t =
   [3.5▼sqrt(2);▼1.5▼9.5]
>> str2num (t)
ans =
      3.5000    1.4142
      1.5000    9.5000
```

يمكن أن تكون أطوال الأسطر غير متساوي

← مصفوفة رمزية

يعيد الايعاز findstr أدلة البداية لسلسلة رمزية موجودة ضمن سلسلة أخرى.

```
>> b = 'Peter Piper picked a peck of pickled peppers';
>> findstr (b, '▼')
ans =
      6   12   1 9   21   26   29   37
>> findstr (b, 'p')
ans =
      9   13   22   30   38   40   41
>> findstr (b, 'cow')
ans =
      [ ]
```

حرف صغير

مصفوفة الخلايا للسلاسل الرمزية

يبدو شرط تساوي عدد الأعمدة في جميع اسطر المصفوفات النصية متعباً، خاصة إذا اختلف عدد الفراغات المضافة من سطر لآخر، ويمكن حل هذه المشكلة عبر استخدام مصفوفات الخلايا، حيث يمكننا وضع كل أنواع البيانات ضمن مصفوفة الخلايا، ويتجلى الاستخدام الأكبر لمصفوفة الخلايا مع السلاسل الرمزية.

تعتبر مصفوفة الخلية ببساطة نوعاً من البيانات التي تسمح للمستخدم بتسمية مجموعة من البيانات ذات الأنواع والحجوم المختلفة، وذلك كما يبينه المثال التالي:

```
>> C = {'How'; 'about'; 'this for a'; 'cell array of strings?'}
C =
    'How'
    'about'
    'this for a'
    'cell array of strings?'
>> size (C)
ans =
      4     1
```

تستخدم أقواس المجموعة { } لإنشاء مصفوفة الخلايا، وذلك استخدمناها في حصر السلسلة الرمزية بأكملها، وتملك المصفوفة C في هذا المثال أربعة اسطر وعموداً واحداً، ويحوي كل عنصر من مصفوفة الخلية سلسلة رمزية مختلفة الطول.

وتعنون مصفوفات الخلايا كما تعنون بقية المصفوفات، وذلك كما يلي:

```
>> C (2: 3)
ans =
    'about'
    'this for a'
>> C ([4   3   2   1])
ans =
    'cell array of strings?'
    'this for a'
    'about'
    'How'
>> C (1)
ans =
    'How'
>> size (C (1))
ans =
    1    1
```

ما زالت النتائج حتى ألان عبارة عن مصفوفات خلايا، وذلك لان التعبير C (indices) يعنون الخلايا المعطاة ولكنه لا يحدد محتوى هذه الخلايا. ولاسترجاع محتويات خلية جزئية محدّدة عليك استخدام أقواس مجموعة كما في المثال التالي:

```
>> S = C {4}
S =
    cell array of strings?
>> size (s)
```

```
ans =
      1   22
```

كما ويمكن عنونة جزء من محتويات مصفوفة الخلية الجزئية كما يلي:

```
>> C {4} (1: 10)
ans =
     cell array
```

يحول الايعاز char محتويات مصفوفة الخلية إلى مصفوفة نصية مناسبة، كما يبينه المثال التالي:

```
>> S = char (C)
S =
    How
     about
     this for a
     cell array of strings?
>> size (S)
ans =
      4     22
```

ويقوم الايعاز cellstr بإجراء التحويل العكسي ويعيد صياغة السلاسل الرمزية بشكل جيد كما يلي:

```
>> cellstr (S)
ans =
      'How'
      'about'
      'this for a'
      'cell array of strings?'
```

جمل الإدخال والإخراج

جمل الإدخال

هناك عدة صيغ للإدخال بالإضافة إلى عملية التنسيب منها:

١- تعليمة input:

مثال (١):

```
>> x = input ('enter x: ')
enter x:
```

مثال (٢): إدخال الأعداد.

```
n = input ('enter n:');
m = input ('enter m:');
for i = 1: n
  for j = 1: m
    result (i, j) = i ^  j;
```

```
  end;
end;
```

مثال (٣): إدخال أسماء رمزية.

```
clc;
clear;
z = input ('enter name', 's');
```

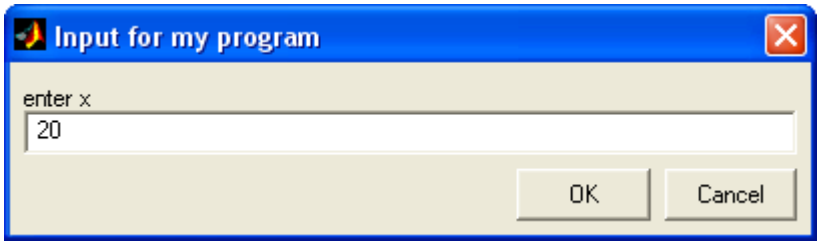
للدلالة على إدخال string

٢- صيغة ثابتة للإدخال (على شكل مربع حوار):

مثال:

```
prompt = {'enter x'};
def = {'20'};
dlgTitle = 'Input for my program';
lineNo = 1;                    % عدد السطور المدخلة
answer = inputdlg (prompt, dlgTitle, lineNo, def);
x = str2num (answer {1});      % تحويل string إلى num في حالة التعامل مع رقم
```

القيمة الأولى من مصفوفة الخلايا



جمل الإخراج

هناك عدة صيغ للإخراج منها:

١- تعليمة disp:

مثال (١):

```
>> d = 15;
>> disp (d);
```

15

مثال (٢):

```
>> a = 'ali';
>> disp (a);
ali
```

مثال (٣):

```
>> sum = 9.8;
>> disp (['sum = ', num2str (sum)]);
sum = 9.8
```

مثال (٤):

```
>> disp ('computer');
computer
```

ملاحظة (١):

يجب أن يكون محتويات disp قيمة ذات نوع بياني واحد ضمن الجملة الواحدة (كل جملة نوع بياني واحد).

ملاحظة (٢):

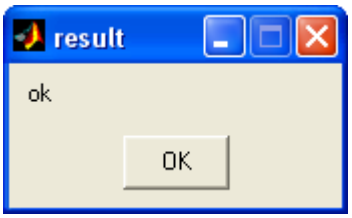
في حالة كون محتويات disp أكثر من قيمة ذات نوع بيانية مختلفة ضمن الجملة الواحدة (يجب ان تجمع القيم في قوسين كبيرين [ ] (مثال (٣))).

٢- تعليمة msgbox:

```
>> msgbox ('ok', 'result')
```

الشيء المطلوب طباعته (نوع بياني رمزي)

عنوان الصندوق



(٣) تعليمة fprintf:

مثال (١):

```
>> y = 1.2;
>> x = 100.5;
>> fprintf ('variable x is % 6.3f\n', x);
>> fprintf ('variable y is % 6.3f\n', y);
variable x is  1.200
variable y is 100.500
```

وهذا يعني بأنه تم حجز 6 مراتب منها 3 مراتب بعد الفارزة العشرية.

مثال (٢):

```
>> fprintf ('% 8.3f\n', round (3.8));
   4.000
```

ملاحظة (١):

يمكن استخدام صيغ مختلفة للطباعة وكما يلي:

%c	رمز واحد
%d	تدوين عشري
%e	تدوين يائي
%f	تدوين النقطة الثابتة
%i	تدوين عشري
%o	تدوين ثماني
%s	تدوين رمزي
%x	تدوين سداسي عشر

ملاحظة (٢):

يمكن طباعة الأعداد والأسماء والنتائج من خلال كتابة الايعازات بدون فارزة منقوطة وستظهر النتائج في نافذة الأمر Command Window.

الجمل الشرطية

يدعم برنامج MATLAB العمليات المنطقية والمقارنة مثلما يدعم العمليات الرياضية، وتهدف العمليات والمعاملات المنطقية الحصول على أجوبة للأسئلة التي يجاب عنها بصح أو خطأ (True/False).

تعتبر لغة MATLAB في تعاملها مع جميع التعابير المنطقية وعمليات المقارنة إن أي عدد غير صفري هو True ويعتبر الصفر False، كما ويكون إخراج جميع التعابير المنطقية وعمليات المقارنة عبارة عن مصفوفات منطقية تحوي العدد واحد من اجل True والعدد صفر من اجل False.

وتعتبر المصفوفات المنطقية نوعاً خاصاً من المصفوفات العددية، كما يمكن عنونة المصفوفة المنطقية بنفس طريقة عنونة باقي المصفوفات التي استخدمها سابقاً ضمن التعابير العددية.

معاملات المقارنة (العوامل العلائقية) : Relational Operators

تتضمن معاملات المقارنة كل الإشارات المقارنة الشائعة والمدرجة في الجدول التالي:

معامل المقارنة	الوصف
<	أصغر من
<=	أصغر أو يساوي
>	أكبر من
>=	أكبر أو يساوي

==	إشارة المساواة (لكي نميزها عن =)
=~	اشارة عدم المساواة

يمكن استخدام معاملات المقارنة للمقارنة بين مصفوفتين لها نفس الحجم، أو للمقارنة بين مصفوفة وعدد مفرد وتتم هذه الحالة مقارنة كل عنصر من المصفوفة مع العدد المفرد، وتكون المصفوفة الناتجة بنفس حجم المصفوفة التي تمت مقارنتها كما يبينه المثال التالي:

مثال (١):

```
>> a = 1; b = 5;
>> x = a > b
x =
    0
>> A = 1: 9, B = 9 - A
A =
    1   2   3   4  5   6   7   8   9
B =
    8   7   6   5   4   3   2   1   0
>> tf = A > 4
tf =
    0   0   0   0  1   1   1   1   1
```

لقد أوجدنا العناصر من A التي هي أكبر من 4، وتظهر الاصفار في المصفوفة الناتجة في مواقع العناصر عندما $A \leq 4$، بينما يظهر الرقم 1 عندما $A > 4$.

```
>> tf  = (A == B)
tf =
    0   0   0   0   0   0   0   0   0
```

لقد تم هنا إيجاد عناصر A التي تساوي العناصر في المصفوفة B.

ملاحظة:

لاحظ بان الإشارتين (=) و (==) تعنيان شيئاً مختلفاً، حيث يقوم (==) بمقارنة متغيرين وتعيد العدد واحد إذا كانا متساويين وصفراً إذا لم يكونا متساويين، بينما تستخدم (=) لإسناد إخراج العملية إلى متغير.

مثال (١): لتوليد مصفوفة أحادية منطقية عناصرها واحدات (في حالة اكبر من thr) واصفاراً (في حالة اصغر من أو تساوي thr).

```
>> inddent = [10   17   22   0   7   3   2];
>> thr = 7;
>> y = (indent > thr)
y =
    1   1   1  0   0    0   0
```

مثال (٢): لتوليد مصفوفة أحادية عناصرها نفس العناصر (في حالة اكبر من thr) واصفاراً (في حالة اصغر من أو تساوي thr).

```
>> z = inddent .* (inddent > thr)
z =
   10   17    22   0   0   0   0
```

المعاملات المنطقية (العوامل المنطقية) :Logical Operators

توفر المعاملات المنطقية طريقة لدمج أو نفي تعابير المقارنة، ويظهر الجدول التالي المعاملات المنطقية الموجودة في لغة MATLAB:

المعامل المنطقي	الوصف
&	AND (و)
|	OR (أو)
~	NOT (نفي)

وسنقدم لك فيما يلي بعض الأمثلة على استخدام المعاملات المنطقية:

```
>> a = 1;
>> b = 5;
>> x = a ~= b
```

```
x =
    1
>> b = (1 == 1) & (2 ~= 3)
b =
    1
>> b = (1==1) | (2 ~= 3)
b =
    1
>> b = (1==1) &  not ((2 ~= 3))
b =
    0
```

```
>> A = 1: 9; B = 9 - A;
>> tf = A > 4
tf =
    0    0    0    0    1    1    1    1    1
```

حيث قام بإيجاد عناصر A التي قيمها اكبر من 4

```
>> tf = ~ (A > 4)
tf =
    1    1    1    1    0    0    0    0    0
```

لقد قام البرنامج بقلب النتيجة السابقة، وتعني استبدال مواقع الاصفار والواحدات.

```
>> tf = (A > 2) & (A < 6)
tf =
    0    0    1    1    1    0    0    0    0
```

حيث تعيد هذه العبارة العدد واحد عندما يكون العنصر من A اكبر من 2 واقل من 6.

أسبقية المعامل

يقوم برنامج MATLAB بإيجاد قيمة تعبير مستنداً إلى مجموعة من القواعد الناظمة لأسبقية المعامل، وتحسب المعاملات ذات الأسبقية العليا قبل المعاملات ذات الأسبقية الدنيا، وتقيم المعاملات ذات الأسبقية المتساوية من اليسار إلى اليمين. ويشرح الجدول التالي قواعد أسبقية المعامل التي يعتدها برامج MATLAB.

المعامل	مستوى الأسبقية
الأقواس ( )	الأعلى
المدور(')، القوة (^، ^.)	
إشارة النفي (~)	
الضرب (*، *.)، القسمة (/، /.)	
الجمع (+)، والطرح (-)	
معامل النقطتين المتعامدتين (:)	
أصغر من (>)، واصغر أو يساوي(=>)، اكبر من (<)، اكبر من أو يساوي(=<)، المساواة (==)،عدم المساواة (=~)	
الجمع المنطقي(&) AND	
المعامل المنطقي (|) OR	الأدنى

الصيغة IF-ELSE-END

قد نحتاج إلى حساب مجموعة من أوامر استناداً إلى إخراج ناتج عن اختبار شرطي. وتنفذ هذه التعليمة في لغة MATLAB عبر استخدام الصيغة if-else-end وكما يلي:

```
if expression
    (commands)
end
```

وستنفذ الأوامر (commands) الواقعة بين العبارتين if وend إذا كانت قيمة التعبير (expression) تكون true. واليك المثال التالي:

```
>> x = 10;
>> if  x == 10
      disp ('ok')
   end;
```

وإذا كان لدنيا خياران، فتصبح الصيغة if-else-end كما يلي:

```
if expression
  (commands evaluated if True)
else
  (commands evaluated if False)
end
```

حيث ستنفذ المجموعة الأولى من الأوامر في حال امتلك التعبير expression القيمة true، بينما تنفذ المجموعة الثانية إذا امتلك التعبير expression القيمة false.

وإذا كانت هناك عدة حالات، فستأخذ التعبير if-else-end الشكل التالي:

```
if expression1
  (commands evaluated if expression1 is true)
elseif expression2
  (commands evaluated if expression2 is true)
elseif expression3
  (commands evaluated if expression3 is true)
elseif expression4
  (commands evaluated if expression4 is true)
.
.
.
```

```
else
  (commands evaluated if no other expression is true)
end
```

واليك الأمثلة التالية:

مثال (١):

```
>> x = 10;
>> if  x == 10
  msgbox ('ok', 'result');
```

مثال (٢):

```
>> if  x ==  10
      msgbox ('ok', 'result');
    else
      msgbox ('no', 'result');
    end;
```

مثال (٣):

```
>> x = 11;
>> if  x == 1
      disp ('1');
    elseif  x == 2
      disp ('2');
    else
      disp ('3');
   end;
```

الإخراج

3

الصيغة SWITCH-CASE

عندما يتوجب علينا تنفيذ أوامر اعتماداً على استخدام متكرر لاختيار كمي لوسط ما، عندها من السهل استخدام الصيغة switch-case التي لها الصيغة العامة التالية:

```
switch expression
  case test-expression1
     (commands1)
  case test-expression2
     (commands2)
   otherwise
       (commands3)
end
```

يجب أن يكون expression هنا أما عدداً مفرداً أو سلسلة رمزية. يقارن التعبير expression الموجود في الصيغة السابقة بالتعبير test-expression1 الموجود في عبارة case الأولى. وإذا تساوى التعبيران، سيتم تنفيذ الأوامر (commands1) وتخطي التعليمات الواقعة بعدها حتى العبارة end. أما إذا لم يتحقق الشرط الأول، فسيختبر الشرط الثاني، حيث سيقارن expression في المثال السابق مع العبارات test-exoression2 الموجودة في عبارة case الثانية. وإذا تساوى التعبيران، سيتم تنفيذ (commands2) وتهمل بقية العبارات حتى عبارة end. إذا لم تحقق أي عبارة case المساواة مع التعبير expression، عندها ستنفذ الأوامر (commands3) التي تلي العبارة otherwise.

لاحظ من الشرح الذي أوردناه عن صيغة switch-case بأنه سيتم تنفيذ أحدى مجموعات الأوامر المكونة للصيغة switch-case واليك الأمثلة التالية:

مثال (١):

```
x = 1;
switch x
   case {1, 2, 3, 4, 5}
      disp ('1..5');
   case {9, 10}
      disp ('9..10');
   otherwise
```

```
    disp ('this is impossible');
end;
```

الإخراج 1..5

مثال (٢):

```
clc;
clear;
n = 3;
switch n
  case {0}
    m = n + 3;
  case {2}
    m = 'ali';
 case {3}
    m = magic (n);
  otherwise
    disp ('error');
end;
disp (m);
```

الإخراج

```
     8     1     6
     3     5     7
     4     9     2
```

مثال (٣):

```
x = 2.7;
units = 'm';
switch units
  case {'inch', 'in'}
```

```
    y = x * 2.54;
  case {'meter', 'm'}
    y = x / 100;
  case {'feed', 'ft'}
    y = x * 2.54 / 12;
  case {'millimeter', 'mm'}
    y = x * 10;
  case {'centimeter', 'cm'}
    y = x;
 otherwise
   disp (['Unknown Units:'    units]);
end;
```

الإخراج

```
y = 0.027
```

جمل الدوران والتكرار

توفر لغة MATLAB مجموعة من جمل الدوران والتكرار وهي:

جملة for

تقوم حلقات for بإعادة تنفيذ مجموعة من الأوامر لعدد معين من المرات وبخطوة معينة، وتعطى الصيغة العامة لحلقة for كما يلي:

```
for  i = x1: x3: x2
  (commands)
end;
```

حيث يعاد تنفيذ الأوامر (commands) الواقعة بين عبارتي for و end من القيمة الابتدائية x1 إلى القيمة النهائية x2 وبزيادة مقدارها x3. كما في المثال التالي:

مثال:

```
>> for n = 1: 10
       x (n) = sin (n * pi / 10);
   end;
>> x
x =
  Columns 1 through 7
    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
  Columns 8 through 10
    0.5878    0.3090    0.0000
```

ويمكن تفسير الدوارة أعلاه كما يلي:

من اجل كل قيمة لـ n من 1 إلى 10 يجب حساب قيمة العبارة الموجودة حتى عبارة end التالية، تكون قيمة n في الدورة الأولى n = 1، وتكون في الدورة الثانية n = 2 وهكذا حتى تصل إلى n = 10.

مثال: توليد 10 أعداد عشوائية قيمتها (1..10).

```
>> array = randperm (10)
array =
      8   2   10   7   4   3   6   9   5   1
>> for n = array
     x (n) = sin (n * pi / 10);
   end;
>> x
x =
   Columns 1 through 7
     0.3090   0.5878   0.8090   0.9511   1.0000   0.9511   0.8090
   Columns 8 through 10
     0.5878   0.3090   0.0000
```

سيأخذ متغير الحلقة n هنا قيماً عشوائية بين (1) و (10) معطاة بالمصفوفة array.

ملاحظة:

يمكن إنشاء عدة حلقات for متداخلة، كما في المثال التالي:

```
>> for n =1: 5
     for m = 5: -1: 1
       A (n, m) = n ^ 2 + m ^ 2;
     end;
     disp (n);
  end;
```

الإخراج

1

2

3
4
5

```
>> A
A =
     2    5    10    17    26
     5    8    13    20    29
     10   13   18    25    34
     17   20   25    32    41
     26   29   34    41    51
```

أمثلة:

```
>> for i = 1: 10
      disp (i);
   end;
```

الإخراج

1
2
3
.
.
10

```
>> for i = 0: 2: 10
disp (i);
   end;
```

الإخراج

0
2

4
6
8
10

```
>> for i = 10: -2: 1
    disp (i);
  end;
```

الإخراج

10
8
6
4
2

(طبع جدول الضرب)

```
>> for i =1: 10
    for j = 1: 10
      mult (i, j) = i * j;
    end;
  end;
```

1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 3 6 40
. . . .
. . . .
10 20 30 40 50 60 70 80 90 100

جملة WHILE

تُجري حلقات while عمليات الحساب عدداً غير محدد من المرات على عكس حلقات for التي تؤدي عدداً معيناً من التمريرات، ويمكن كتابة الصيغة العامة لحلقة while كما يلي:

```
while expression
      (commands)
end;
```

ستنفذ مجموعة الأوامر(commands) الواقعة بين العبارتين while و end طالما أن كل العناصر ضمن expression تمتلك قيماً صحيحة (true)، وعادةً ما تكون نتيجة expression عدداً مفرداً.

مثال (١):

```
>> x = 1;
>> while x < 25
     disp (x);
= x + 1;
    end;
```

x

الإخراج

```
1
2
3
.
.
24
```

مثال (٢):

```
>> num = 0; EPS = 1;
>> while (1 + EPS) > 1
     EPS = EPS / 2;
     num = num + 1;
   end;
>> num
num =
      53
```

ملاحظة:

هناك طريقة قانونية للخروج من حلقة for وwhile وكالاتي:

(في حال تحقق الشرط يتم الخروج من الدوارة for وكذلك while)

```
s = 0;
x = 1;
while x < 100
   s = s + x;
   if  s > 250
      break;
   end;
   x = x + 5;
end;
```

الإخراج

x = 51

s = 286

```
s = 0;
for i = 1: 100
   s = s + i;
   if s > 250
      break;
   end;
end;
```

الإخراج

i = 22

s = 253

ملاحظة:

أذا وجدت التعليمة break ضمن حلقة داخلية واقعة ضمن حلقات اكبر فان البرنامج يخرج من الحلقة التي صادف فيها التعليمة ولا يخرج من الحلقات الأكبر.

ملفات البيانات الخاصة ببرنامج MATLAB

يمكن تخزين المتغير الموجود في ساحة عمل برنامج MATLAB، وفق صيغة خاصة ببرنامج MATLAB، وذلك عن طريق استخدام الأمر save كما يلي:

```
>> save
```

وبذلك يتم خزن جميع المتغيرات الموجودة في ساحة العمل (Workspace) في ملف ذي صيغة ثنائية باسم matlab.mat يوضع في المجلد الحالي. وتحافظ هذه الملفات ذات صيغة الثنائية، والخاصة ببرنامج MATLAB، على كامل القيم وبدقة مضاعفة، كما وتخزن أسماء المتغيرات بنفس الدقة، ولا تعتبر ملفات MAT-files ذات أصول مستقلة، إنما هي متوافقة تماماً مع بقية أنواع الملفات الموجودة في برنامج MATLAB، إذ نستطيع تخزين أي متغير وفق نوع من الملفات وفتحة من قبل الأنواع الأخرى دون إجراء أية معالجة خاصة للملف.

ويمكن أن يستخدم الأمر save لتخزين متغيرات معينة كما في المثال التالي:

```
>> save   var1   var2   var3
```

أي قم بتخزين المتغيرات var1 وvar2 و var3 ضمن الملف matlab.mat، ويمكن أن نحدد اسم الملف كوسيط أول للأمر save كما يلي:

```
>> save   filename   var1   var2   var3
```

وتفسر التعليمة السابقة كما يلي: خزّن المتغيرات var1, var2, var3 ضمن ملف اسمه filename.mat.

ويعاكس الأمر load الأمر save إذ يفتح هذا الأمر ملفات البيانات التي تم إنشاؤها بالأمر save كما يلي:

```
>> load
```

وهي تعني حملّ كل المتغيرات التي تجدها ضمن الملف matlab.mat حيثما وجدته أولا سواءً في المجلد الحالي أو في مسار البحث لبرنامج MATLAB. ويتم تخزين أسماء المتغيرات المخزونة في الملف matlab.mat في ساحة العمل، وستحمّل فوق المتغيرات ذات الأسماء المطابقة لها في حال وجودها.

ولتحميل متغيرات معينة من ملف ذي لاحقة (MAT-file) يجب ان نذكر اسم الملف وقائمة المتغيرات كما يلي:

```
>> load   filename   var1, var2, var3
```

لقد تم هنا فتح الملف filename.mat وحملت المتغيرات var1, var2, var3 إلى ساحة العمل.

ايعازات المجموعات والبتات والايعازات القاعدية

ايعازات المجموعات

نستطيع تقييم المصفوفات على إنها مجموعات لأنها تجميع منتظم لعدد من القيم وانطلاقاً من هذا الفهم، يقدّم لك برنامج MATLAB عدة توابع لاختبار ومقارنة المجموعات، ويقدم لك المثال التالي ابسط اختبار للمساواة:

```
>> a = rand (2, 5);
>> b = rand (2, 5);
>> isequal (a, b)
ans =
     0
>> isequal (a, a)
ans =
     1
```

ويقدم لك المثال التالي الايعاز unique بحذف العناصر المتكررة من وسط الإدخال:

```
>> a = [2: 2: 8; 4: 2: 10]
a =
   2   4   6   8
   4   6   8   10
>> unique (a)
ans =
     2
     4
     6
     8
    10
```

ويمكن تحديد مجموعة العناصر المشتركة بين وسيطين عبر استخدام الايعاز ismember كما يلي:

```
>> a = 1: 9
a =
```

```
    1    2    3    4    5    6    7    8    9
>> b = 2: 2: 9
b =
    2    4    6    8
>> ismember (a, b)
ans =
      0    1    0    1    0    1    0    1    0
>> ismember (b, a)
ans =
      1    1    1    1
```

كذلك الايعاز union لاتحاد مجموعتين.

```
>> union (a, b)
ans =
      1    2    3    4    5    6    7    8    9
```

كذلك إيعاز intersect لتقاطع مجموعتين.

```
>> intersect (a, b)
ans =
      2    4    6    8
```

كذلك إيعاز setdiff للفضلة بين مجموعتين.

```
>> setdiff (a, b)
 ans =
      1    3    5    7    9
```

ملاحظة:

يمكن إجراء العمليات السابقة على مصفوفات رمزية أو مصفوفات خلايا.

إيعاز البت

إضافة إلى المعاملات المنطقية التي ذكرناها سابقاً، يؤمن البرامج توابعاً تسمح بإجراء العمليات المنطقية على بتات منفصلة من الأعداد الصحيحة.

```
>> bitand (3, 4)
ans =
      0
>> bitor (3, 4)
ans =
      7
>> bitxor (13, 27)
ans =
      22
>> bitcmp (20, 5)          متمم العدد ٢٠ لخمس بتات
ans =
     11
 >> bitset (30, 1)        جعل البت الأولى من ٣٠ يكون ١
ans =
      31        الموقع
>> bitget (30, 1)          جلب البت الأولى من ٣٠
ans =
      0        العدد   الموقع
>> bitshift (3, 2)         إزاحة لليسار (موقعين)
ans =
     12
 >> bitshift (12, -2)       إزاحة لليمين (موقعين)
ans =
     3
>> z = [7    5    4  ;  3    8    9];
>> circshift (z, 1)
ans =
```

```
    3   8   9
    7   5   4
```

الايعازات القاعدية

يؤمن برنامج MATLAB العديد من الأوامر التي تحول الأعداد العشرية إلى قواعد أخرى وفق صيغ سلاسل رمزية ونستطيع التحويل بين الأعداد العشرية والأعداد الثنائية عبر الايعازين bin2dec، dec2bin كما يلي:

```
>> a = dec2bin (17)
ans =
     10001
>> class (a)
ans =
     char
>> bin2dec (a)
ans =
     17
>> class (ans)
ans =
     double
```

عدد الخانات

```
>> dec2bin (17, 6)
ans =
    01001
```

ويتم التحويل بين الأعداد العشرية والستة عشرية (يكون أساس العد فيها العدد 16) عبر الايعازين hex2dec و dec2hex كما يلي:

```
>> a = dec2hex (2047)
a =
   7FF
```

عدد الخانات

```
>> dec2hex (2047, 4)
```

```
ans =
      07FF
>> class (a)
ans =
      char
>> hex2dec (a)
ans =
      2047
>> class (ans)
ans =
      double
```

رمزي

الدوال والبرامج الفرعية

تستخدم الدوال بشكل واسع في لغة MATLAB. والصيغة العامة للدوال هي:

المخرجات المدخلات

```
[out1, out2,…, outn] = function_name (input1, input2,...,inputn);
```

اسم الدالة

كما في الأمثلة التالية:

مثال (١):

البرنامج الرئيسي

```
>> x = [1, 2, 3, 4, 5, 6, 7, 8];
>> y = [11, 12, 13, 2, 9, 70];
>> avgx = average1 (x);
>> avgy = average1 (y);
```

البرنامج الفرعي (الدالة)

```
function result = average1 (z);
  L = length (z);
  sum1 = sum (z);
   result = sum1 / L;
```

```
>> avgx
avgx =
        4.5000
>> avgy
avgy =
        19.5000
```

مثال (٢):

الإدخال الإخراج

البرنامج الرئيسي

```
>> res1 = mult2 (x);
>> res2 = mult2 (y);
```

البرنامج الفرعي (الدالة)

```
function result = mult2 (x);
  result = 2 * x;
```

```
>> res1
res1 =
     2    4    6    8    10    12    14    16
>> res2
res2 =
    22    24    26    4    18    140
```

مثال (٣):

```
>> [sin_x, cos_x, x_2] = multf (x);
>> [sin_y, cos_y, y_2] = multf (y);
```

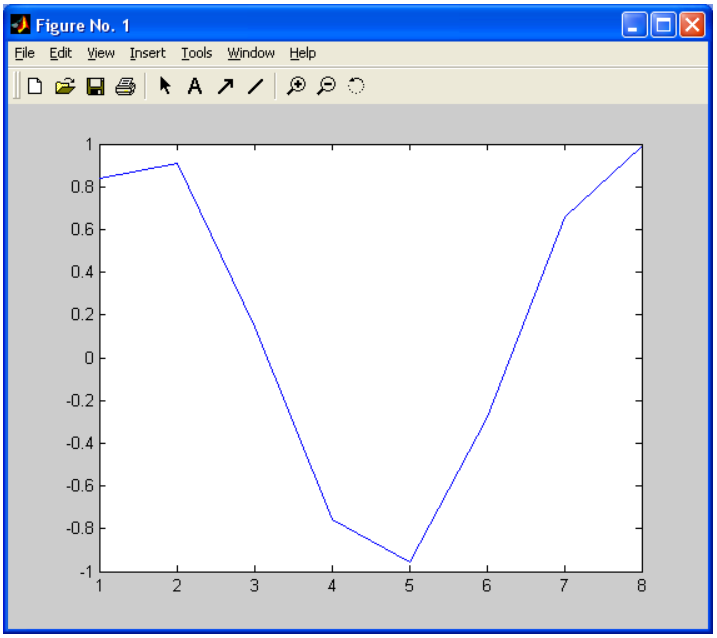
البرنامج الرئيسي

```
function [x1, x2, x3] = multf (x);
  x1 = sin (x);
  x2 = cos (x);
  x3 = 2 * x;
```

البرنامج الفرعي (الدالة)

```
>> plot (sin_x)
```

لرسم النقاط



ملاحظات:

١- الدالة التي تكون على شكل ملف مفصول لايمكن تنفيذها مباشرة إلا بعد استدعائها بالبرنامج الرئيسي.

٢- تكون اسم الدالة المخزون في القرص الصلب واسم الدالة بعد علامة (=) الموجود في السطر الأول يجب ان يكونا متطابقين.

مثال:

```
function  y = myfunction (a, b);
-------
-------
-------
```

فيكون الخزن myfunction.m

٣- لا تحتوي نهاية الدالة على (end).

٤- تخزن البرامج بعد كل تعديل وخاصة الدوال.

٥- اسم الدالة المخزون يجب أن تبدأ بحرف.

٦- يمكن أن تستدعي دالة من قبل دالة أخرى.

الرسوم البيانية

يزودك برنامجMATLAB بالعديد من الايعازات التي تظهر البيانات ثنائية الأبعاد وثلاثية الأبعاد، حيث يرسم بعضها منحنيات ثنائية الأبعاد وثلاثية الأبعاد بينما يرسم بعضها سطوحاً وإطارات، كما يمكن استخدام اللون كبعد رابع.

الايعاز plot

يقوم هذا الايعاز بإظهار البيانات على شكل ثنائي الأبعاد.

مثال (١):

```
x = [1: 0.5: 10];
```

y = exp (x); لاحتساب مجموعة قيم للـy (مصفوفة)

plot (x, y) x, y لرسم قيم بيانية للمحورين

مثال (٢):

```
x = 1: 10;
plot (x)
```

ملاحظة:

في حالة وجود إحداثي واحد (قائمة واحدة) يقوم الايعاز plot برسم قيم بيانية متناظرة بالمحورين أي (x, x) لكل عناصر القائمة.

مثال (٣):

```
y = [ ];
for i = 1: 10
   y (i) = exp (i);
 end;
  plot (y);
```

مثال (٤):

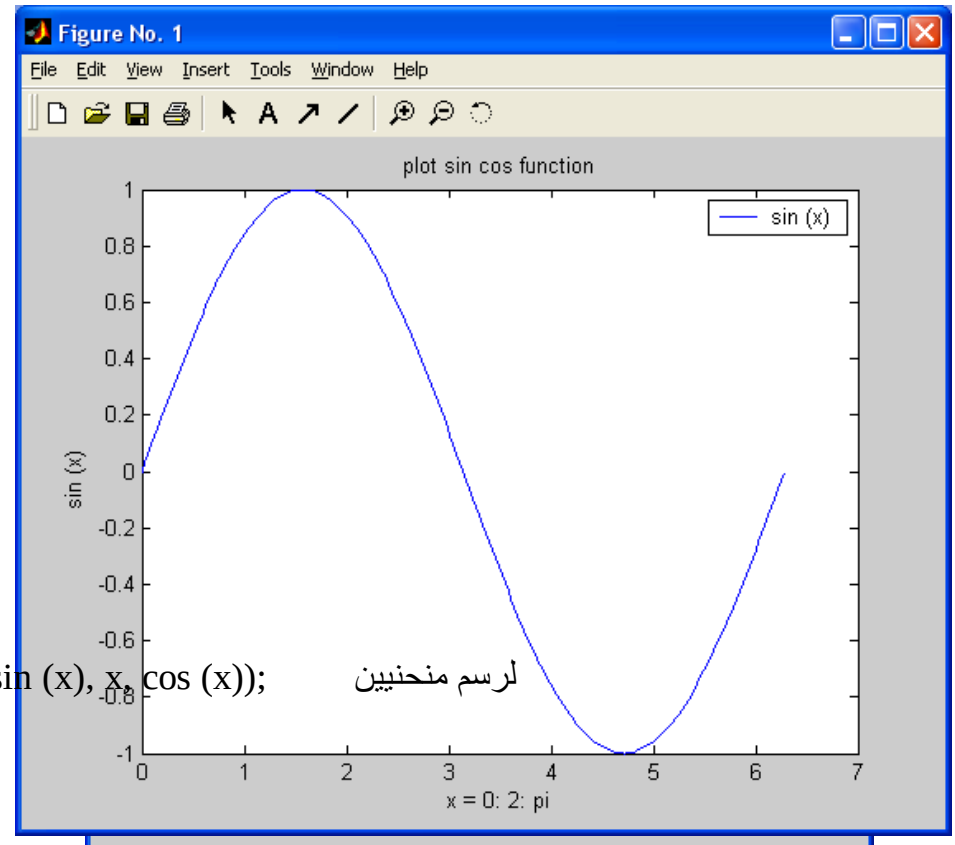
```
y = [ ];
for i = 1: 10
   y = [y exp (i)];
end;
plot (y);
```

مثال (٥):

ارسم مخطط بياني (graph).

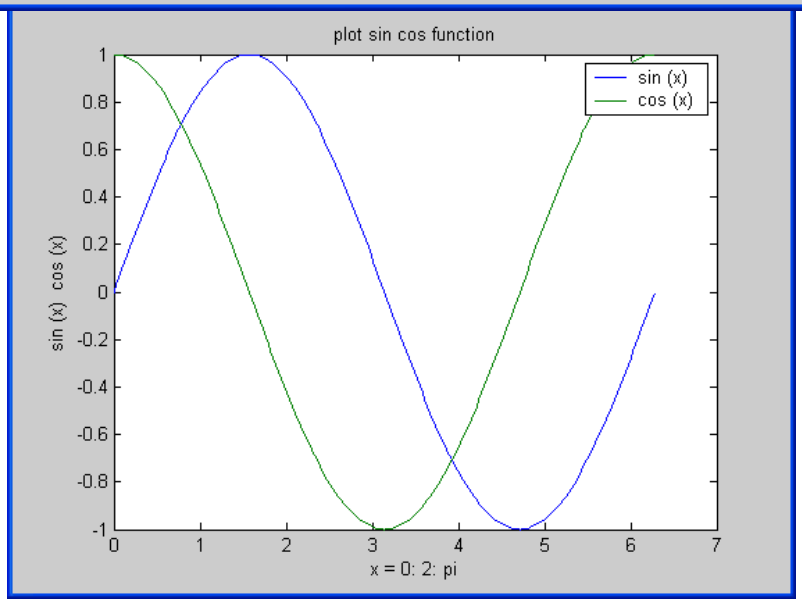
```
clc;
clear;
x = 0: pi / 100: 2 * pi;
y = sin (x);
```

```
plot (x, y);
legend ('sin (x)');          دليل المخطط
xlabel ('x = 0: 2: pi');          عنوان المحور x
ylabel ('sin (x)  cos (x)');          عنوان المحور y
title ('plot sin cos function');      عنوان المخطط الرئيسي (أعلى المخطط)
```



مثال (٦):

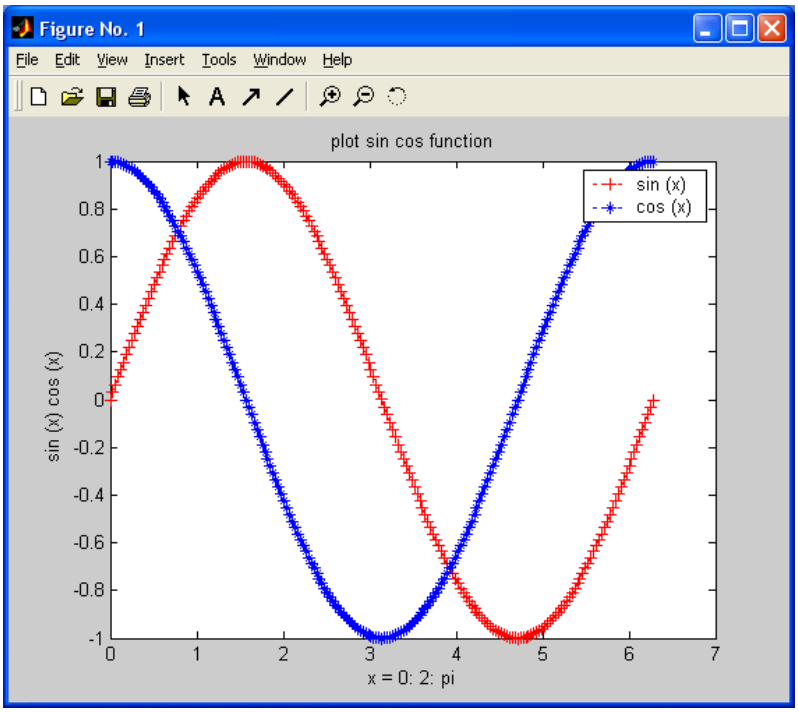
```
plot (x, sin (x), x, cos (x));          لرسم منحنيين
```



مثال (٧):

```
plot (x, sin (x), 'r: +', x, cos (x), 'b: *');
```

لون احمر — علامة المخطط sin (x) — لون ازرق — علامة المخطط cos (x)



ملاحظة:

يمكن كتابة أي نص على المخطط باستخدام الايعاز:

```
text (x, y, 'string');
```

الاحداثي السيني — الاحداثي الصادي — النص المطلوب كتابته

إيعاز plot3

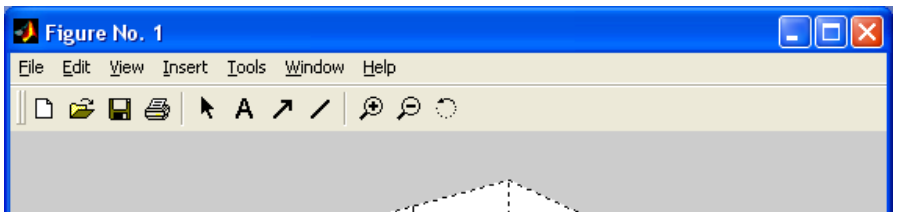
لقد تم تمديد الايعاز plot إلى ثلاثي الأبعاد وأصبح plot3، وصيغته لها نفس صيغة plot ثنائي البعد عدا كون البيانات لها ثلاث مساقط بدلاً من مسقطين. والصيغة العامة لها:

```
plot3 (x1, y1, z1, s1, x2, y2, z2, s2,…);
```

الاحداثي السيني — الاحداثي الصادي — الاحداثي الثالث — اللون (خيط رمزي)

مثال:

```
t = linspace (0, 10 * pi, 100);
plot3 (sin (t), cos (t), t);
```



```
xlabel ('sin (t)');
ylabel ('cos (t)');
zlabel ('t');
text (0, 0, 0, 'origin');
grid on          لرسم الشبكة
```

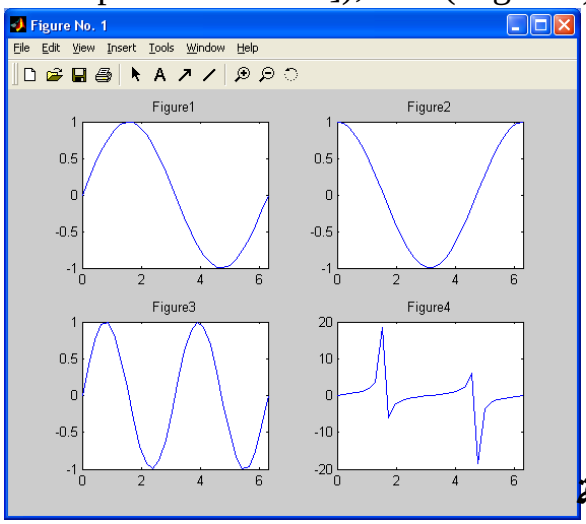
الرسوم البيانية الجزئية

تستطيع نافذة figure واحدة ان تمسك باكثر من مجموعة محاور أو صور، حيث يقسم subplot (m, n, p) نافذة الشكل الحالية الى مصفوفة m*n لرسم المناطق ويختار المساحة p لتصبح فعالة. لقد رسمت الرسومات البيانية الجزئية من اليسار الى اليمين وعلى طول الصف العلوي، ثم على طول الصف السفلي وهكذا، وذلك كما يلي:

مثال:

```
x = linspace (0, 2 * pi, 30);
y = sin (x);
z = cos (x);
a = 2 * sin (x) .* cos (x);
b = sin (x) ./ (cos (x) + eps);
subplot (2, 2, 1);
plot (x, y); axis ([0    2 * pi      -1        1]); title ('Figure1');
subplot (2, 2, 2);
plot (x, z); axis ([0     2 * pi      -1        1]); title ('Figure2');
```

```
subplot (2, 2, 3);
plot (x, a); axis ([0      2 * pi      -1          1]); title ('Figure3');
subplot (2, 2, 4);
plot (x, b); axis ([0      2 * pi      -20       20]); title ('Figure4');
```

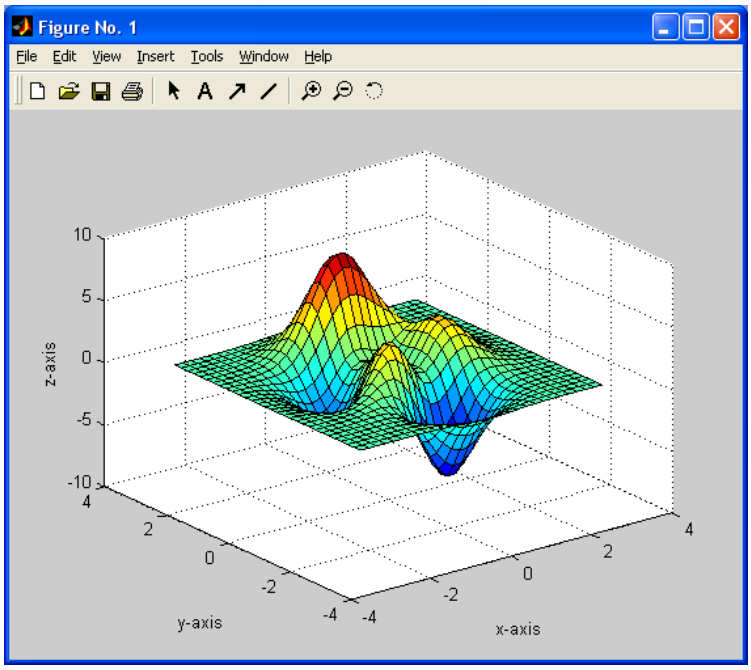


الرسوم البيانية السطحية

تشبه الرسوم البيانية السطحية تلك الرسوم البيانية عدا انها تعبر عن المساحات الواقعة، عبر استخدام الايعاز surf كما يلي:

مثال (١):

```
[x    y    z] = peaks (30);
surf (x, y, z);
xlabel ('x-axis');
ylabel ('y-axis');
zlabel ('z-axis');
```



مثال (٢):

```
for i = 1: 10
  for j =1: 10
    mult (i, j) = i * j;
  end;
end;
surf (mult)    شكل مجسم (ثلاثي الابعاد)
```

ملاحظة:

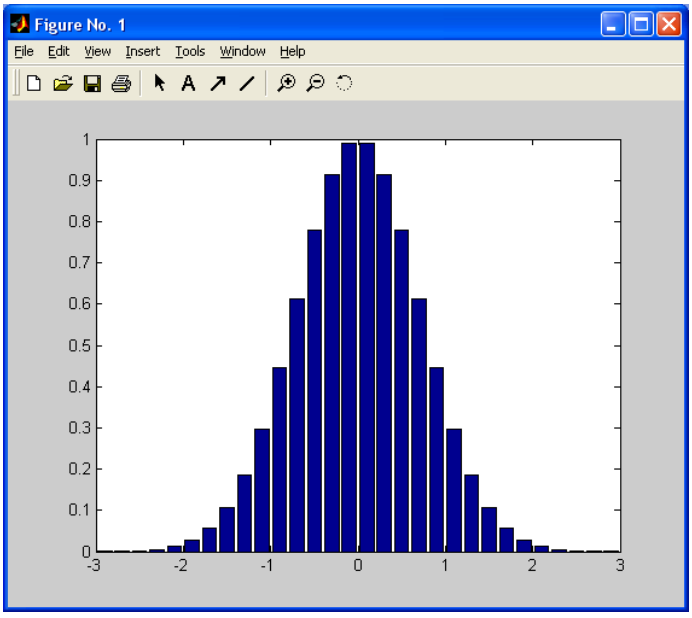
هناك من الايعازات لرسم أشكال هندسية منها:

الايعاز bar

يستخدم لرسم bar chart

مثال:

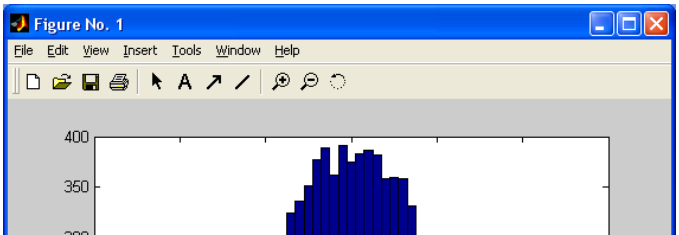
```
x = -2.9: 0.2: 2.9;
bar (x, exp (-x .* x));
```



الايعاز hist

يستخدم لرسم histogram

مثال:



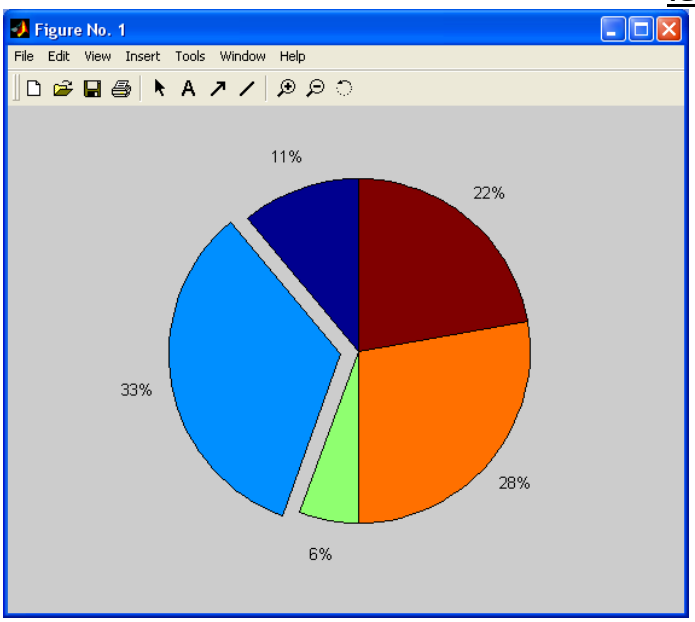
```
x = -2.9: 0.1: 2.9;
y = randn (10000, 1);
hist (y, x);
```

الايعاز pie

يستخدم لرسم pie chart

مثال:

```
x = [1    3    0.5    2.5    2];
explode = [0    1    0    0    0];
pie (x, explode);
```



مثال: لرسم مخطط بياني.

```
clear;
clc;
corr = [0.0012, 0.0208, 0.0633, 0.1391];
```

```
amount = [1, 2, 3, 4];
subplot (211);
plot (amount, corr, '--rs');
title ('Cipher-image  VS  Amount of Encrypted Data');
xlabel ('Amount of Encrypted Data');
ylabel ('Cipher-image Correlation');
```